## **Left Triangle Pattern**

| import java.util.Scanner; public class Main {  public static void main(String args[]) {   Scanner sc = new Scanner(System.in);  int n = sc.nextInt(); //i for rows and j for columns  //row denotes the number of rows you want to print  // int i, j, row = 6;  //Outer loop work for rows  for (int i=1; i<=n; i++) {  //inner loop work for space  for (int j=2\*(n-i); j>=1; j--) {  //prints space between two stars  System.out.print(" ");  }  //inner loop for columns  for (int j=1; j<=i; j++ ) {  //prints star  System.out.print("\* ");  }  //throws the cursor in a new line after printing each line  System.out.println();  }  }  } |
| --- |

## **Calculate nPr**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int r = sc.nextInt();  int npr = 1;  for(int i=1; i<=r; i++){  npr = npr\*n;  n--;  }  System.out.println(npr);  } } |
| --- |

# **Calculate nCr**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);    //System.out.println();  int n = sc.nextInt();  int r = sc.nextInt();    int ncr = 1;  // int npr = 1;  for(int i=1; i<=r; i++){  ncr = ncr\*n/i;  // npr = npr\*n;  n--;  }  System.out.println(ncr);  // System.out.println(npr);    } } |
| --- |

## **Second Method**

| import java.util.\*;  public class Main {  static long factorial(int n) {  long res = 1;  for(int i = 2; i<=n; i++) {  res \*= i;  }  return res; }  static long calculate\_nCr(int n, int r) {  long ans = factorial(n) / ( factorial(r) \* factorial(n-r) );  return ans; }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  int n = scanner.nextInt();  int r = scanner.nextInt();  long ans = calculate\_nCr(n,r);  System.out.print(ans); } } |
| --- |

## **Count frequency Of Number**

| import java.util.\*; public class Main{ static int FrequencyofDigits(long n, int d){  // Example -> n = 158686888, d = 8   // answer -> 5   int count = 0;  while(n>0){  long lastDigit = n%10l;  if(lastDigit == d){  count = count + 1;  }  n = n/10l;  }  return count;  }  public static void main(String[] args){  Scanner sc = new Scanner(System.in);  long n = sc.nextLong();  int d = sc.nextInt();  System.out.println(FrequencyofDigits(n, d));  } } |
| --- |

## **Frequency OF Digit**

| import java.util.\*; public class Main{ static int FrequencyofDigits(long n, int d){  // write code here  int count = 0;  while(n>0){  long lastDigit = n%10l;  if (lastDigit == d)  {  count = count + 1;  }  n=n/10l;  }  return count;  }  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  long n = sc.nextLong();  int d = sc.nextInt();  System.out.println(FrequencyofDigits(n, d));  } } |
| --- |

## **Place Value Checker**

| import java.util.\*; public class Main{  public static void main(String[] args){  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  sc.close();  boolean res = determineSecondLastDigit(n);  if (res)  System.out.println("Yes");  else  System.out.println("No");  }  public static boolean determineSecondLastDigit(int n){  n = n/10;  int lastDigit = n%10;   if(lastDigit == 0){  return true;  }else{  return false;  }  } } |
| --- |

## **Binary to Decimal**

| import java.io.\*; import java.util.\*; public class Main { public static int binToDec(String s)  {  //Write code here  int dec\_value = 0;  int base = 1;  int len = s.length();  for (int i = len - 1; i >= 0; i--){  if (s.charAt(i) == '1')  dec\_value += base;  base = base \* 2;  }   return dec\_value; }  public static void main(String args[]) throws IOException{  Scanner sc = new Scanner(System.in);  int t = sc.nextInt();  while(t > 0){  String s = sc.next();  int ans = binToDec(s);  System.out.println(ans);  t -= 1;  }   } } |
| --- |

## **Diamond Pattern Stars**

| import java.util.Scanner; public class Main {  public static void main(String arg[])   {  Scanner in = new Scanner(System.in);  int t = in.nextInt();  for(int k=1; k<=t; k=k+1)  {  int n = in.nextInt();  int spaces = n-1;  int stars = 1;  for(int i=1; i<=n; i=i+2)  {  for(int j=1; j<=spaces; j=j+1)  {  System.out.print(" ");  }  for(int j=1; j<=stars; j=j+1)  {  System.out.print("\* ");  }  System.out.println();  spaces = spaces - 2;  stars = stars + 2;  }  spaces = 2;  stars = n-2;  for(int i=1; i<=(n/2); i=i+1)  {  for(int j=1; j<=spaces; j=j+1)  {  System.out.print(" ");  }  for(int j=1; j<=stars; j=j+1)  {  System.out.print("\* ");  }  System.out.println();  spaces = spaces + 2;  stars = stars - 2;  }  }  } } |
| --- |

#### Second Approach

| import java.util.\*;  class Solution {  public void pattern(int row\_size) {  for (int out = row\_size; out >= -row\_size; out--) {  for (int in1 = 1; in1 <= Math.abs(out); in1++) {  System.out.print(" ");  }  for (int in2 = row\_size; in2 >= Math.abs(out); in2--)  System.out.print("\* ");  System.out.println();  }  } }  public class Main {   public static void main(String[] args) throws Throwable {  Scanner cs = new Scanner(System.in);  int n = cs.nextInt();  cs.close();  Solution solution = new Solution();  solution.pattern(n);  } } |
| --- |

**THIRD APPROACH**

| import java.util.\*;  public class Main {   public static int[] SumArrayExpectSelf(int[] nums, int n) {  //Write code here  int totalSum = 0;  for(int i=0; i<n; i++) {  totalSum = totalSum + nums[i];  }  int ans[] = new int[n];  for(int i=0; i<n; i++) {  ans[i] = totalSum-nums[i];  }  return ans;  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int[] nums = new int[n];   for(int i = 0; i < n; i++) {  nums[i] = sc.nextInt();  }   int[] Ans = SumArrayExpectSelf(nums, n);   for(int a : Ans)  System.out.print(a + " ");  } } |
| --- |

## **Stopwatch**

| import java.io.\*; import java.util.\*;  public class Main {  public static void main(String args[]) {  Scanner input = new Scanner(System.in);  int n = input.nextInt();  int a[] = new int[n];  for(int i = 0; i < n; i++){  a[i] = input.nextInt();  }  Solution s = new Solution();  System.out.println(s.stopwatch(a));  } }  class Solution {  static String stopwatch(int[] a) {  int n = a.length;  if(n % 2 != 0) return "still running";  int sum = 0;  for(int i = 0; i < n; i += 2){  sum += a[i + 1] - a[i];  }    return Integer.toString(sum);  } } |
| --- |

## **Staircase Pattern**

| import java.util.Scanner; import java.io.\*;  public class Main{  public static void main(String[] args) {  Scanner scn = new Scanner(System.in);  int n = scn.nextInt();  //write code here  for(int i=1; i<=n; i++){  for(int j=1; j<=(n-i); j++){  System.out.print(" ");//This is for Spaces  }  for(int j=1; j<=i;j++){  System.out.print("#");  }  System.out.println();  }  } } |
| --- |

## **Print Continuous Character Pattern**

| import java.util.\*;  public class Main {   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  sc.close();  // write code here  for(int i=1; i<=n; i++){  char currentCharacter = (char)(65+i-1);  for(int j=1; j<=i; j++){  System.out.print(currentCharacter);  currentCharacter++;  if(currentCharacter>'Z'){  currentCharacter='A';  }  }  System.out.println();  }  } } |
| --- |

## **Optimus Prime**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner in = new Scanner(System.in);  // end = 15   int end = in.nextInt(); // ending point     // STEP 1, i=2, 2<=15, 2 will be printed   // STEP 2, i=3, 3<=15, 3 will be printed  // STEP 3, i=4, 4<=15, 4 will not be printed   // STEP 5, i=5, 5<=15, 5 will be printed   // STEP 6, i=6, 6<=15, 6 will not be printed   for(int i=2; i<=end; i=i+1)  {  // we will check whether i has any divisor from 2 to sqrt(i)    // If i is prime, will it have any divisor from 2 to sqrt(n) ?  boolean isPrime = true;    // i = 2, Math.sqrt(2) = 1.414 ~ 1 (integer part)  // i = 3, Math.sqrt(3) = 1.713 ~ 1 (integer part)  // i = 4, Math.sqrt(4) = 2.0 ~ 2 (integer part)  // i = 5, Math.sqrt(5) = 2. ~ 2 (integer part)  // i = 6, Math.sqrt(6) = 2. ~ 2 (integer part)  for(int j=2; j<=(int)Math.sqrt(i); j=j+1)  {  if(i%j == 0)  {  isPrime = false;  break;  }  }    if(isPrime == true)  {  System.out.println(i);  }  }    // Fact -> Divisors they exist in pairs   // 36 --> 1 2 3 4 6 9 12 18 36     // For all composite numbers, there will be a divisor which   // be less than sqrt(n)    // Prime number (n) --> 1 and n    // Will prime number have any divisor from 2 to sqrt(n) ?  } } |
| --- |

### **Armstrong Number In Range**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner in = new Scanner(System.in);  int m = in.nextInt(); // starting point   int n = in.nextInt(); // ending point   // STEP 1 We need to go from one point to another point (For loop)  for(int i=m; i<=n; i=i+1){  // STEP 2 For every number we need to check it is Armstrong or not  // for every number i we need tell it is Armstrong or not   // STEP 3 How many digits are there in that number ?   int countDigits = 0;  int copyOfI = i; // If you change the value of i then outer loop will be spoiled   // copyOfI = 100  // STEP (i), 100>0, countDigits = 0 + 1 = 1, copyOfI will become 10   // STEP (ii), 10>0, countDigits = 1 + 1 = 2, copyOfI will become 1   // STEP (iii), 1>0, countDigits = 2 + 1 = 3, copyOfI will become 0   while(copyOfI>0){  countDigits = countDigits + 1;   copyOfI = copyOfI/10; // removing the last digit   }  // STEP 4 We need to raise every digit to that power and add it  copyOfI = i;   int sum = 0; // 153   // STEP (i), 153>0, lastDigit = 3, sum = 0 + 3^3 = 27, copyOfI will become 15   // STEP (ii), 15>0, lastDigit = 5, sum = 27 + 5^3 = 152, copyOfI will become 1  // STEP (iii), 1>0, lastDigit = 1, sum = 152 + 1^3 = 153, copyOfI will become 0   while(copyOfI>0{  int lastDigit = copyOfI%10;  sum = sum + (int)Math.pow(lastDigit, countDigits);  copyOfI = copyOfI/10;   }  // STEP 5 We need to check if original is as same as the sum then it is Armstrong number  // STEP 6 If it is Armstrong number then we print it  if(sum == i){  System.out.print(i + " ");  }  }  } } |
| --- |

### **Character Pattern**

| import java.util.\*; public class Main {  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  // Write your code here  int alphabet = 65;  for(int i=1; i<=n; i++){  for(int j=1; j<=i; j++){  System.out.print((char) alphabet);  }  alphabet++;  System.out.println();  }  } } |
| --- |

### **Right Angle Triangle Stars**

| import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  for(int i=1; i<=n; i++){  for(int j=1; j<=i; j++){  System.out.print("\*");  }  System.out.println();  }  } } |
| --- |

### **Print Number Pattern 2**

| import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  sc.close();  for(int i=1; i<=n; i++){  for(int j=i; j>=1; j--){  System.out.print(j + "");  }   System.out.println();  }  } } |
| --- |

### **Star Pyramid**

| import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  for(int i=1; i<=n; i++){  for(int j=1; j<=(n-i); j++){  System.out.print(" ");  }  for(int k=1; k<=i; k++){  System.out.print("\* ");  }  System.out.println();  }  }  } |
| --- |

### **Grading System**

| import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n= sc.nextInt();  sc.close();  if(n>90){  System.out.println("Excellent");  }else if(n>80 && n<=90){  System.out.println("Good");  }else if(n>70 && n<=80){  System.out.println("Fair");  }else if(n>60 && n<=70){  System.out.println("Meets Expectations");  }else{  System.out.println("Below Expectations");  }  } } |
| --- |

### **Leap Year**

| import java.util.\*; public class Main {  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int year = sc.nextInt();  int isleap=1;  if(year%4==0){  if(year%100==0){  if(year%400!=0){  isleap=0;  }else{  isleap=1;  }  }else{  isleap=1;  }  }else{  isleap=0;  }  System.out.println(isleap);  } } |
| --- |

### **Which Angle Tringle**

| import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int a= sc.nextInt();  int b= sc.nextInt();  int c= sc.nextInt();  sc.close();  int max\_side = a;  if(max\_side < b)  max\_side = b;  if(max\_side < c)  max\_side = c;  if(2 \* max\_side \* max\_side == a \* a + b \* b + c \* c)  System.out.println(2);  else if(2 \* max\_side \* max\_side < a \* a + b \* b + c \* c)  System.out.println(1);  else  System.out.println(3);  } } |
| --- |

### **Big Light**

| import java.util.\*; public class Main {  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int h1 = sc.nextInt();  int h2 = sc.nextInt();  int v1 = sc.nextInt();  int v2 = sc.nextInt();  if(v1==v2){  System.out.println("false");  }else if((h1-h2)%(v2-v1)==0){  System.out.println("true");  }else{  System.out.println("false");  }  } } |
| --- |

### **Which Case**

| import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  char a = sc.next().charAt(0);  sc.close();  if(a>= 'A' && a<='Z'){  System.out.println(1);  }else if(a>='a' && a<='z'){  System.out.println(0);  }else{  System.out.println(-1);  }  } } |
| --- |

### **Verify Cube**

| import java.util.\*; public class Main { public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  long a = sc.nextLong();  long b = sc.nextLong();  sc.close();  long lhs = (a+b)\*(a+b)\*(a+b);  long rhs = (a\*a\*a)+(b\*b\*b)+(3\*a\*b\*b)+(3\*a\*a\*b);  System.out.println(lhs);  System.out.println(rhs);  if(lhs==rhs){  System.out.println("VERIFIED");  }else{  System.out.println("NOT VERIFIED");  }  } } |
| --- |

### **Conditional Problem 1**

| import java.io.\*; import java.util.\*; public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  if(n==28){  System.out.println("i am young");  }else{  System.out.println("i am not young");  }  } } |
| --- |

### **Conditional Problem 2**

| import java.util.\*; import java.lang.\*; import java.io.\*; public class Main{  public static void main (String[] args) throws java.lang.Exception{  Scanner sc =new Scanner(System.in);  int a;  a=sc.nextInt();  if(a<30){  System.out.println("less important");  }else{  System.out.println("more important");  }  } } |
| --- |

### **Conditional Problem 3**

| import java.util.\*; public class Main { public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  if(n>1){  System.out.println("You entered more");  }else{  System.out.println("You entered less");  }  } } |
| --- |

### **Conditional Problem 5**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner in = new Scanner(System.in);  int n = in.nextInt();  if(n%6 == 0){  System.out.println("Divisible");  }else{  System.out.println("Not divisible");  }  }  } |
| --- |

### **Conditional Problem 6**

| import java.util.Scanner; public class Main {  public static void main(String arg[]) {  Scanner in = new Scanner(System.in);  int a = in.nextInt();  int b = in.nextInt();   if(a%2 == 1 && b % 2 == 1 ) {  System.out.println("we are odd");  } else {  System.out.println("we are simple");  }  }  } |
| --- |

### **Quadrant Question Code (Mystery Rooms and Chambers)**

| import java.util.Scanner; public class Main {  public static void main(String arg[]) {  Scanner in = new Scanner(System.in);  int x = in.nextInt();  int y = in.nextInt();  if(x>0 && y>0) {  System.out.println("1");  } else if(x<0 && y>0) {  System.out.println("2");  } else if(x<0 && y<0){  System.out.println("3");  } else {  System.out.println("4");  }  } } |
| --- |

### 

### **Sum and Product of Digits of a Number**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  int num, sum = 0, prod = 1, rem;  Scanner in = new Scanner(System.in);  //System.out.println();  num = in.nextInt();  while(num>0){  rem = num % 10;  sum = sum + rem;  prod = prod \* rem;  num = num / 10;  }  System.out.println(sum+prod);  //System.out.println(prod);  } } |
| --- |

### 

### **Sam's Favourite Number**



| import java.util.\*; class Accio {  public void FavouriteNumber(int[] arr, int k) {  for(int i=arr.length-1; i>=0; i--) {  if(arr[i] == k)  System.out.print(i + " ");  }  } }  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int k = sc.nextInt();   int[] arr = new int[n];   for(int i = 0; i < n; i++) {  arr[i] = sc.nextInt();  }  Accio Obj = new Accio();  Obj.FavouriteNumber(arr, k);  } } |
| --- |

### **Number Pyramid**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  for(int i=1; i<=n; i++){   for(int j=1; j<=n-i; j++){  System.out.print(" ");  }  for(int j=1; j<=i; j++){  System.out.print(i + " ");  }  System.out.println();  }  } } |
| --- |

### **Number Of Days Question**

| import java.util.\*;  public class Main {  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int month = sc.nextInt();  if(month == 2) {  System.out.println("28");  } else if(month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12) {  System.out.println("31");  }else  System.out.println("30");  } } |
| --- |

### **Sum Of Natural Number**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  long N = sc.nextLong();  // scanner.close();  System.out.println((N\*(N+1))/2);  } } |
| --- |

### **HCF of Two Numbers**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int a = sc.nextInt();  int b = sc.nextInt();  int hcf = 1;  for(int i = 1; i<=Math.min(a,b); i++){  if(a%i==0 && b%i==0){  hcf=i;  }  }  System.out.println(hcf);  } } |
| --- |

### **Check Prime Number**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  boolean prime=true;  if(n<2){  System.out.println(n + " is not a prime number");  return;  }  for(int i=2; i\*i<=n; i++){  if(n%i==0){  System.out.println(n + " is not a prime number");  return;  }  }  System.out.println(n + " is a prime number");  sc.close();  } } |
| --- |

### **Odd Number**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int n=0, count=0;  while(n%2!=1){  n=sc.nextInt();  count++;  }  System.out.println(count);  } } |
| --- |

### **Reverse Number**

| import java.util.Scanner; public class Main {  public static void main(String arg[]) {  Scanner in = new Scanner(System.in);  int n = in.nextInt();  int reversedNumber = 0;  while(n>0) {  int lastDigit = n%10;   reversedNumber = reversedNumber\*10 + lastDigit;  n = n / 10;  }  System.out.println(reversedNumber);  } } |
| --- |

### **Power Of A Number**

| import java.util.\*; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int a = sc.nextInt();  int b = sc.nextInt();  sc.close();  int power = 1;  for (int i = 1; i <= b; i++)  power = power \* a;  System.out.println(power);  } } |
| --- |

### 

### **Palindrome Checker: Verifying if a Number Reads the Same Backwards and Forwards**

| import java.util.\*;  public class Main { public static void main(String args[]) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();   String s = String.valueOf(n);  int l = 0, r = s.length() - 1;  boolean ok = true;  while(l < r){  if(s.charAt(l) != s.charAt(r)){  ok = false;  break;  }  l++;  r--;  }if(ok == true) {  System.out.println("true");  } else {  System.out.println("false");  }  } } |
| --- |

### **Sum Of Digit**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner in = new Scanner(System.in);  int n = in.nextInt();  int sum = 0;  while(n>0){  int lastDigit = n%10;  sum = sum + lastDigit;  n = n/ 10;  }  System.out.println(sum);  } } |
| --- |

### **Even Sum**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  long n = sc.nextLong();  long sum = 0;  for(int i=1; i<=n; i++)  if(i%2==0){  sum+=i;  }  System.out.println(sum);  sc.close();  } } |
| --- |

### **Factorial With Loop**

| import java.util.\*;  public class Main{  public static void main(String arg[]){  Scanner input = new Scanner(System.in);  int n = input.nextInt();  input.close();  long ans = 1;  for(int i = 2; i <= n; i++){  ans\*=i;  }  System.out.println(ans);  } } |
| --- |

### **Array Problem 1(Ruma and Sima)**

| import java.util.\*;  public class Main {  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int arr[] = new int[n];  for(int i=0;i<n;++i){  arr[i] = sc.nextInt();  }  int ans=ArrayProblem1(n,arr);  System.out.println(ans);  }   public static int ArrayProblem1(int n, int[] arr){  int largestElement = arr[0];  for(int i=0; i<n; i=i+1)  {  if(arr[i]>largestElement)  {  largestElement = arr[i];  }  }  int indexOfLargestElement = 0;  for(int i=0; i<n; i=i+1)  {  if(largestElement == arr[i])  {  indexOfLargestElement = i;  break;  }  }  return indexOfLargestElement;  } } |
| --- |

### **Array Problem 2 (Write a program to count the elements in an array which are greater than 35)**

| import java.util.\*;  public class Main {  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int arr[] = new int[n];  for(int i=0;i<n;++i){  arr[i] = sc.nextInt();  }  int ans=ArrayProblem2(n,arr);  System.out.println(ans);  }  public static int ArrayProblem2(int n, int[] arr){  int count = 0;  for(int i=0; i<n; i=i+1)  {  if(arr[i]>35)  {  count = count + 1;  }  }  return count;  } } |
| --- |

### **Index Of Element**

| import java.util.\*;  public class Main {  public static void IndexOfElement(int N, int X, int[] arr){      boolean isXPresent = false;    for(int i=0; i<N; i=i+1)  {  if(arr[i] == X)  {  isXPresent = true;    System.out.print(i+1+" ");  }  }    if(isXPresent == false)  {  System.out.print("-1");  }  }  public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);  int N = sc.nextInt();  int X = sc.nextInt();   int arr[]=new int[N];  for(int i = 0; i < N; i++)  {  arr[i] = sc.nextInt();  }  IndexOfElement(N,X,arr);  } } |
| --- |

### 

### **Array Operations**

**Given an array arr of size n, Write a program to find the below 3 values.**

1. **Sum of all the elements in the array.**
2. **Average of all the elements in the array (Give the floor value).**
3. **Maximum element in the array.**

| import java.util.\*; public class Main {  public static void main(String[] args) {    Scanner scanner = new Scanner(System.in);    int N = scanner.nextInt();     int[] arr = new int[N];  for(int i = 0; i < N; i++)  arr[i] = scanner.nextInt();   ArrayOperations(arr, N);  }   public static void ArrayOperations(int[] arr, int N) {  int sum=0, max=arr[0];   for(int i = 0; i < N; i++) {  sum+=arr[i];  max=Math.max(max, arr[i]);  }   System.out.println(sum+" "+(sum/N)+" "+max);  } } |
| --- |

### **Plus Minus**

| import java.util.\*;   public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int[] arr = new int[n];  for (int i = 0; i < n; i++)  arr[i] = sc.nextInt();  printRatios(arr);  sc.close();  }   public static void printRatios(int[] arr) {  // write code here  double positive = 0, negative =0, zero =0;  for(int i=0; i<arr.length; i++)  {  if(arr[i]>0)  {  positive++;  }  else if(arr[i]<0)  {  negative++;  }  else  {  zero++;  }  }  //convert array length into doubledouble l = arr.length;  double l = arr.length;   double p = positive/l;  double n = negative/l;  double z = zero/l;  System.out.println(String.format("%.6f",p));  System.out.println(String.format("%.6f",n));  System.out.println(String.format("%.6f",z));  } } |
| --- |

### **GST Calculator**

| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  float purchaseAmount = sc.nextFloat();  float gst = 0;  if (purchaseAmount <= 5000) {  gst = purchaseAmount \* 18 / 100;  } else if (purchaseAmount > 5000 && purchaseAmount <= 10000) {  gst = purchaseAmount \* 20 / 100;  } else if (purchaseAmount > 10000 && purchaseAmount <= 20000) {  gst = purchaseAmount \* 25 / 100;  } else {  gst = purchaseAmount \* 28 / 100;  }  System.out.println(String.format("%.3f", gst));  } } |
| --- |

### **Alternate Manner Matrix Traversal**

| import java.util.\*;  class Solution {  public void printElementsAlternately(int[][] mat,int m,int n) {  //Write code here and print output  for(int i=0; i<m; i=i+1) // rows   {  if(i%2 == 0) // even row (LEFT TO RIGHT)  {  for(int j=0; j<n; j=j+1)  {  System.out.print(mat[i][j] + " ");  }  }  else // ODD row (RIGHT TO LEFT)  {  for(int j=n-1; j>=0; j=j-1)   {  System.out.print(mat[i][j] + " ");  }  }  }  } }   public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int m, n;  m = sc.nextInt();  n = sc.nextInt();  int[][] mat = new int[m][n];  for (int i = 0; i < m; i++)  for (int j = 0; j < n; j++)  mat[i][j] = sc.nextInt();  Solution Obj = new Solution();  Obj.printElementsAlternately(mat,m,n);    sc.close();  } } |
| --- |

### **Array Problem 4**

**Write a program to find out the difference between sum of all the elements in two arrays arr1 and arr2.**

**If sum of all elements in arr1 is larger then, the output should be First array is larger,**

**if sum of all elements in arr2 is larger then, the output should be Second array is larger,**

**otherwise the output should be Both are equal.**

| import java.io.\*; import java.util.\*;   public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;    n = sc.nextInt();  int []arrA=new int[n];  for(int i=0;i<n;++i){  arrA[i]=sc.nextInt();  }    n = sc.nextInt();  int []arrB=new int[n];  for(int i=0;i<n;++i){  arrB[i]=sc.nextInt();  }    System.out.println(ArrayProblem(arrA,arrB));   }  public static String ArrayProblem(int []arrA,int []arrB){  int n = arrA.length;   int difference = 0;  for(int i = 0; i < n; i++){  difference += arrA[i] - arrB[i];  }  if(difference == 0) return "Both are equal";  return ((difference > 0) ? "First array is larger":"Second array is larger");  } } |
| --- |

### **Array Problem 5**

Karan has an array arr and an integer k. He wants to find out the number of consecutive pairs of array elements whose sum will be k.

Write a program to count the number of consecutive pairs.

Note that, if arr[5]={1,2,3,4,5} then consecutive pairs are (1,2),(2,3),(3,4),(4,5).

**Input Format**

**The first line contains n, the size of the array, and a positive integer k.**

**The second line contains n elements of the array.**

| import java.io.\*; import java.util.\*;  public class Main {   public static int ArrayProblem(int []arr,int k){  // Write code here  int n = arr.length;  int count = 0;  for(int i=0; i<n-1; i=i+1)  {  if(arr[i]+ arr [i+1] == k)    {  count = count + 1;  }  }  return count;  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n,k;  n = sc.nextInt();  k = sc.nextInt();  int []arr=new int[n];  for(int i=0;i<n;++i){  arr[i]=sc.nextInt();  }  System.out.println(ArrayProblem(arr,k));   } } |
| --- |

### **Pairs Problem**

Given an array of N integers, count the total pairs of integers that have a difference of K.

**Input Format**

1st line contains N & K (integers).

2nd line contains N numbers of the array.

| import java.io.\*; import java.util.\*;  class Solution {  public int Pairs(int []arr,int k){  //your code here  int n = arr.length;     // Math.abs(-3) = 3   // Math.abs(5) = 5     // Math.abs() --> If we give it a negative, it will give you positive value of the number     // arr = [1, 5, 3, 4, 2]   // k = 2, n = 5  int count = 0;    // STEP 1, i=0, fixed element -> arr[0], j will run 4 times   // STEP 2, i=1, fixed element -> arr[1], j will run 3 times   // STEP 3, i=2, fixed element -> arr[2], j will run 2 times   // STEP 4, i=3, fixed element -> arr[3], j will run 1 times   // STEP 5, i=4, fixed element -> arr[4], j will run 0 times     for(int i=0; i<n-1; i=i+1)  {  for(int j=i+1; j<n; j=j+1)  {  // this is when i = 0  // (i) i = 0, j = 1, arr[0] - arr[1]  // (ii) i = 0, j = 2, arr[0] - arr[2]  // (iii) i = 0, j = 3, arr[0] - arr[3]  // (iv) i = 0, j = 4, arr[0] - arr[4]      // this is when i = 1  // (i) i = 1, j = 2, arr[1] - arr[2]  // (ii) i = 1, j = 3, arr[1] - arr[3]  // (iii) i = 1, j = 4, arr[1] - arr[4]  if(Math.abs(arr[i] - arr[j]) == k)  {  count = count + 1;  }  }  }    return count;    } }   public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n,k;  n = sc.nextInt();  k = sc.nextInt();  int []arr=new int[n];  for(int i=0;i<n;++i){  arr[i]=sc.nextInt();  }  Solution Obj = new Solution();  System.out.println(Obj.Pairs(arr,k));   } } |
| --- |

### **Find Split Point**

Write a program to Partition an array into two subarrays with the same sum. Take input of size of array and the array elements then find if it is possible to split the array in two subarrays (without reordering the elements), such that the sum of the two subarrays is the same.

**Input Format**

Input consists of two lines.

First line contains an integer N denoting size of array

Second line contains N integers denoting the array elements separated by single space

| import java.io.\*; import java.util.\*;  public class Main {  static void findSplitPoint(int arr[], int n) {  //Write your code here  int greenBoxSum = 0;  for(int i=0; i<n; i=i+1){  greenBoxSum = greenBoxSum + arr[i];  }  boolean isPartitionFound = false;  int redBoxSum = 0;  for(int i=0; i<n-1; i=i+1){  redBoxSum = redBoxSum + arr[i];  int blueBoxSum = greenBoxSum - redBoxSum;  if(redBoxSum == blueBoxSum){  isPartitionFound = true;  for(int k=0; k<=i; k=k+1){  System.out.print(arr[k] + " ");  }  System.out.println();  for(int k=i+1; k<n; k=k+1){  System.out.print(arr[k] + " ");  }  }  }  if(isPartitionFound == false){  System.out.println("Not Possible");  }  }  public static void main(String[] args) throws Exception {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int arr[] = new int[n];  for (int i = 0; i < n; i++) {  arr[i] = sc.nextInt();  }  findSplitPoint(arr, n);   } } |
| --- |

### **Toeplitz Matrix**

Given an m x n matrix, return true if the matrix is Toeplitz. Otherwise, return false.

A matrix is Toeplitz if every diagonal from top-left to bottom-right has the same elements.

**Input Format**

First line contains two integers, m and n which define the row and column size of the matrix respectively.

Next each of 'm' lines contain 'n' space separated integers that denote the matrix elements.

| import java.io.\*; import java.util.\*;  class Solution {  public boolean isToeplitzMatrix(int[][] matrix, int m, int n) {  // write code here    // m -> rows   // n -> columns     // j = n-1, j+1 = n (Will this be present in the matrix)    boolean isToeplitzMatrix = true;    for(int i=0; i<m-1; i=i+1)  {  for(int j=0; j<n-1; j=j+1)  {  if(matrix[i][j] != matrix[i+1][j+1])  {  isToeplitzMatrix = false;  break;  }  }  }    return isToeplitzMatrix;    } }  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int m, n;  m = sc.nextInt();  n = sc.nextInt();  int[][] A = new int[m][n];  for(int i = 0; i < m; i++) {  for(int j = 0; j < n; j++) {  A[i][j] = sc.nextInt();  }  }  Solution Obj = new Solution();  boolean ans = Obj.isToeplitzMatrix(A,m,n);  System.out.println(ans);  } } |
| --- |

### **Z traversal of transpose matrix**

Alice loves patterns and recently she came across the 'Z' pattern where she tries to visualize objects in the shape of english capital alphabet 'Z'. But she found it difficult to create or visualize the 'Z' pattern in a matrix. More specifically, she wanted to find the 'Z' traversal of the transpose matrix but was facing difficulties doing the same. Help Alice find the 'Z' traversal of the transpose matrix i.e. 'Z' traversal of the transpose of the matrix.

Note: The 'Z' traversal of the matrix is the traversal of matrix in the shape of 'Z' i.e. moving in the top row from left to right, then moving across the secondary diagonal and finally moving along the last row from left to right.

Complete the function ZTranspose which receives n and input matrix as parameters and prints the 'Z' traversal of the transpose matrix.

**Input Format**

The first line of input contains a single integer n representing the size of the square matrix.

The next each of n lines contains n space-separated integers representing the elements of the matrix.

| import java.util.\*;   class Main {    public static void ZTranspose(int n, int[][] arr) {  int[][] temp = new int[n][n];  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++)  temp[i][j] = arr[j][i];  }  for (int i = 0; i < n - 1; i++) {  System.out.print(temp[0][i] + " ");  }  int r = 0, c = n - 1;  while (r < n) {  System.out.print(temp[r][c] + " ");  r++;  c--;  }  for (int i = 1; i < n; i++) {  System.out.print(temp[n - 1][i] + " ");  }  }    public static void main(String[] args) throws Exception {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int[][] arr = new int[n][n];  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  arr[i][j] = sc.nextInt();  }  }  sc.close();  ZTranspose(n, arr);  }  } |
| --- |

### 

### **Sam's Favourite Number**

Sam was given an array arr of size n and asked to pick any elements from arr. Sam also has a favourite number k. He wants to find all the indices of his favourite number in the array.

Your task is to help Sam by printing the indices of his favourite number in descending order.

If you do not find his favourite number, then nothing is to be printed.

The array is 0-based indexing.

**Input Format**

First line contains two integers n and k which are the size of arr and favourite number, respectively. Second line contains n space separated integers of array arr

| import java.util.\*; class Accio {  public void FavouriteNumber(int[] arr, int k) {  for(int i=arr.length-1; i>=0; i--) {  if(arr[i] == k)  System.out.print(i + " ");  }  } }  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int k = sc.nextInt();   int[] arr = new int[n];   for(int i = 0; i < n; i++) {  arr[i] = sc.nextInt();  }   Accio Obj = new Accio();  Obj.FavouriteNumber(arr, k);  } } |
| --- |

### **Array :- Two Sum**

Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

| class Solution {  public int[] twoSum(int[] nums, int target) {  Map<Integer, Integer>map=new HashMap<>();  for(int i=0; i<nums.length; i++) {  int cur = nums[i];  int x = target-cur;  if(map.containsKey(x)){  return new int[]{map.get(x), i};  }  map.put(cur, i);  }  return null;  } } |
| --- |

### **Array:- Add Two Numbers**

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

| class Solution {  // Add Two Numbers (Java improved)  public ListNode addTwoNumbers(ListNode l1, ListNode l2) {  ListNode dummyHead = new ListNode(0);  ListNode curr = dummyHead;  int carry = 0;  while (l1 != null || l2 != null || carry != 0) {  int x = (l1 != null) ? l1.val : 0;  int y = (l2 != null) ? l2.val : 0;  int sum = carry + x + y;  carry = sum / 10;  curr.next = new ListNode(sum % 10);  curr = curr.next;  if (l1 != null)  l1 = l1.next;  if (l2 != null)  l2 = l2.next;  }  return dummyHead.next;  } } |
| --- |

### **Plus Minus**

Given an array arr of N integers, Calculate the ratios of its elements that are positive, negative, and zero to the array size N.

Print the decimal value of each fraction in a new line up to 6 decimal places.

**Input Format**

The first line contains an integer N, the size of the array.

The second line contains N space-separated integers representing the array arr.

| import java.util.\*;   public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int[] arr = new int[n];  for (int i = 0; i < n; i++)  arr[i] = sc.nextInt();  printRatios(arr);  sc.close();  }  public static void printRatios(int[] arr) {  // write code here  double positive = 0, negative =0, zero =0;  for(int i=0; i<arr.length; i++) {  if(arr[i]>0) {  positive++;  } else if(arr[i]<0) {  negative++;  } else {  zero++;  }  }  //convert array length into doubledouble l = arr.length;  double l = arr.length;   double p = positive/l;  double n = negative/l;  double z = zero/l;  System.out.println(String.format("%.6f",p));  System.out.println(String.format("%.6f",n));  System.out.println(String.format("%.6f",z));  } } |
| --- |

### **Diagonal Differences**

Given a square matrix of size N, calculate the absolute difference between the sums of its diagonals.

**Input Format**

The first line contains a single integer N, the number of rows and columns in the square matrix .

Each of the next N lines describes a row, and consists of N space-separated integers .

| import java.io.\*; import java.util.\*; import java.text.\*; import java.math.\*; import java.lang.\*;; import java.util.regex.\*;  public class Main {   public static void main(String[] args) {  Scanner in = new Scanner(System.in);  int n = in.nextInt();  int a[][] = new int[n][n];  int primaryDiagSum = 0;  int secondaryDiagSum = 0;  for(int a\_i=0; a\_i < n; a\_i++){  for(int a\_j=0; a\_j < n; a\_j++){  a[a\_i][a\_j] = in.nextInt();    if(a\_i == a\_j){  primaryDiagSum += a[a\_i][a\_j];  }    if(a\_i+1+a\_j+1 == n+1){  secondaryDiagSum += a[a\_i][a\_j];  }   }  }  System.out.println(Math.abs(primaryDiagSum - secondaryDiagSum));  } } |
| --- |

**Rough Code**

| import java.io.\*; import java.util.\*;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n, m;  n = sc.nextInt();  m = sc.nextInt();  int [][]mat=new int[n][m];  for(int i=0;i<n;++i){  for(int j=0;j<m;++j){  mat[i][j]=sc.nextInt();  }  }   spirallyTraverse(mat);   System.out.println('\n');  }   public static void spirallyTraverse(int [][]matrix) {    int n = matrix.length; // rows   int m = matrix[0].length; // columns     int top = 0; // top -> topmost row  int bottom = n-1; // bottom -> bottommost row   int left = 0; // left -> leftmost column  int right = m-1; // right -> rightmost column      while(top<=bottom && left<=right)  {  // 1. Print topmost row (FIRST ARROW)  boolean didFirstArrowRun = false;    for(int i=left; i<=right; i=i+1)  {  didFirstArrowRun = true;  System.out.print(matrix[top][i] + " ");  }    // 2. Print rightmost column (SECOND ARROW)  if(didFirstArrowRun == true)  {  boolean didSecondArrowRun = false;    for(int i=top+1; i<=bottom; i=i+1)  {  didSecondArrowRun = true;  System.out.print(matrix[i][right] + " ");  }    // 3. Print bottommost row (THIRD ARROW)  if(didSecondArrowRun == true)  {  boolean didThirdArrowRan = false;  for(int i=right-1; i>=left; i=i-1)  {  didThirdArrowRan = true;  System.out.print(matrix[bottom][i] + " ");  }    // 4. Print leftmost column (FOURTH ARROW)  if(didThirdArrowRan == true)  {  for(int i=bottom-1; i>=top+1; i=i-1)  {  System.out.print(matrix[i][left] + " ");  }  }  }  }    top = top + 1;  bottom = bottom - 1;   left = left + 1;   right = right - 1;  }  } } |
| --- |

### **Roman To Integer**

| import java.util.\*; class Solution { public int romanToInt(String s) { HashMap<Character,Integer>map=new HashMap<Character,Integer>();   map.put('I',1);  map.put('V',5);  map.put('X',10);  map.put('L',50);  map.put('C',100);  map.put('D',500);  map.put('M',1000);  int result=map.get(s.charAt(s.length()-1));  for(int i=s.length()-1;i>0;i--)  {  if(map.get(s.charAt(i))<=map.get(s.charAt(i-1)))  {  result+=map.get(s.charAt(i-1));  }  else  {  result-=map.get(s.charAt(i-1));  }  }  return result; } } |
| --- |

### **Bob's Prime Array**

Alice and Bob are playing a game where Bob gives Alice an array arr of size n. There are some positions in the array where the index (According to 1 based indexing ) and the element at that index both are prime numbers.

Alice has to find those elements and give that to Bob in order of their appearence in array.

Your task is to help Alice complete his task and print the elements.

NOTE: Complete the function and print the elements in the function itself. Print -1 if there are no such elements.

**Input Format**

First line contains an integer n representing the number of elements in the array that bob gave.

Second line contains n space-separated elements of the array arr.

| import java.util.\*;  public class Main {  public static void sieveOfEratosthenes(int n, boolean[] prime)  {  for (int i = 0; i <= n; i++)  prime[i] = true;    for (int p = 2; p \* p <= n; p++) {  if (prime[p] == true) {  for (int i = p \* p; i <= n; i += p)  prime[i] = false;  }  }  }    public static void BobsPrimeArray(int n, int[] arr) {  boolean prime[] = new boolean[10001];  sieveOfEratosthenes(10000,prime);  boolean ansFound = false;  for(int i=1;i<n;++i){  if(arr[i]!=1 && prime[i+1] && prime[arr[i]]){  System.out.print(arr[i] + " ");  ansFound = true;  }  }  if(ansFound == false) System.out.println(-1);  }   public static void main(String[] args) {  Scanner scn = new Scanner(System.in);  int n = scn.nextInt();  int[] arr = new int[n];  for(int i=0;i<n;++i){  arr[i] = scn.nextInt();  }  BobsPrimeArray(n,arr);  } } |
| --- |

### **A Contest**

A Contest is held and a total of N participants took part in the contest. You are already given the scores of the participants in the form of an array (of size N).

The contestant who has a score greater than or equal to the score of the K-th Participant (where K<=N) will advance to the next round, as long as the contestant has a positive score.

Calculate the total number of participants who will advance to the next round.

**Input Format**

The first line of the input contains two integers N and K separated by a single space.

The second line contains N space-separated integers representing the array.

| import java.io.\*; import java.util.\*;  public class Main {  public static void printFindParticipants(int[] arr, int n, int k) {  int count = 0;  for(int i=0; i<n; i=i+1) {  if(arr[i]>=arr[k-1] && arr[i]>0) {  count = count + 1;  }  }  System.out.println(count);  }    public static void main(String[] args) {  Scanner sc = new Scanner(System.in);   int n= sc.nextInt();  int k=sc.nextInt();  int[] arr= new int[n];   for(int i=0;i<n;i++)  arr[i]=sc.nextInt();    printFindParticipants(arr,n,k);  sc.close();    } } |
| --- |

### **2nd Largest from array**

Given an unsorted array of size N with distinct elements. Find the 2nd largest element from the array without sorting the array.

**Input Format**

The first line contains a single integer N.

The second line consists of N integers of the array.

| import java.io.\*; import java.util.\*;  public class Main {  public static void SecondLargest(int[] arr, int n) {  // sorted array -> array is in ascending order or in descending order     // ascending order this means element are in increasing order     // 1 2 3 5 6 -> This is an sorted array   // 6 5 4 3 2 1 -> This is an sorted array     // 1 2 4 3 6 5 -> Unsorted array means array is present in random order   // 6 1 5 7 1 5 2 -> Unsorted array     // If current element is > largest element   // 2nd largest element = largest element   // largest element = current element     // If current element > 2nd largest element but less than largest element  // 2nd largest element = current element    // Integer.MIN\_VALUE -> A big negative number -> -2147483648    // arr = [-6, -7, -9, -5, -8]  // Can we assume largestElement = 0 ? No   // We can assume largestElement = -inf, as all will be greater than that     // Let's assume array is [4, 1, 2, 7]  int largestElement = Integer.MIN\_VALUE;  int secondLargestElement = Integer.MIN\_VALUE;    // STEP 1: i=0, arr[i] = 4, arr[i]>largestElement (true), secondLargestElement = -ve, largestElement = 4   // STEP 2: i=1, arr[i] = 1, arr[i]>largestElement (false), arr[i]>secondLargestElement (true), secondLargestElement = 1   // STEP 3: i=2, arr[i] = 2, arr[i]>largestElement (false), arr[i]>secondLargestElement (true), secondLargestElement = 2  // STEP 4: i=3, arr[i] = 7, arr[i]>largestElement (true), secondLargestElement = 4, largestElement = 7  for(int i=0; i<n; i=i+1)  {  if(arr[i]>largestElement)  {  secondLargestElement = largestElement;  largestElement = arr[i];  }  else if(arr[i]>secondLargestElement)  {  secondLargestElement = arr[i];  }  }    System.out.println(secondLargestElement);  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);   int n= sc.nextInt();  int[] arr= new int[n];   for(int i=0;i<n;i++)  arr[i]=sc.nextInt();    SecondLargest(arr,n);  sc.close();    } } |
| --- |

Second Approach

| import java.io.\*; import java.util.\*;   public class Main {  public static void SecondLargest(int[] arr, int n) {  // Write code here  int max, secondMax;  max = arr[0];  for(int i=0; i<n; i++) {  if(arr[i]>max) {  max = arr[i];  }  }  secondMax = Integer.MIN\_VALUE;  for(int i=0; i<n; i++){  if(arr[i]!=max && arr[i]>secondMax){  secondMax = arr[i];  }  }  System.out.println(secondMax);  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);   int n= sc.nextInt();  int[] arr= new int[n];   for(int i=0;i<n;i++)  arr[i]=sc.nextInt();    SecondLargest(arr,n);  sc.close();    } } |
| --- |

Third Approach

| import java.io.\*; import java.util.\*;   public class Main {  public static void SecondLargest(int[] arr, int n) {  // Write code here  int max1, max2;  if(arr[0]>=arr[1]) {  max1=arr[0];  max2=arr[1];  }else {  max1= arr[1];  max2=arr[0];  }  for(int i=1; i<n; i++) {  if(arr[i]>max1 && arr[i]>max2){  max2=max1;  max1=arr[i];  } else if(arr[i]>max2)  max2=arr[i];  }  System.out.println(max2);  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);   int n= sc.nextInt();  int[] arr= new int[n];   for(int i=0;i<n;i++)  arr[i]=sc.nextInt();    SecondLargest(arr,n);  sc.close();    } } |
| --- |

### **Reverse an array**

You are given an array arr of length n. You have to print the reversed array.

**Input Format**

The first line of input contains an integer n, size of the array.

The next line contains n space-separated integers denoting the elements of the array.

| import java.io.\*; import java.util.\*;   public class Main {   public static void reverseArray(int arr[], int start, int end)  {  // REVERSE array   // WHY we are not printing the array because main function is handling the printing part     int i = start; // start = 0 (starting index of the array)  int j = end; // end = n-1 (ending index of the array)    // arr = [1, 2, 3, 4, 5] , n = 5    // STEP 1, i = 0, j = 4, 0<4 (true) arr[i] = 1, arr[j] = 5,  // STEP 2, i = 1, j = 3, 1<3 (true) arr[i] = 2, arr[j] = 4  // STEP 3, i = 2, j = 2, 2<2 (false), The loop will stop  while(i<j)  {  // we will interchange/swap the values  int temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;    i = i+1; // i will move forward  j = j-1; // j will move backward   }  }    public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int arr1[] = new int[n];  for(int i=0;i<n;i++)  arr1[i] = sc.nextInt();    reverseArray(arr1, 0, n-1);    for (int i = 0; i < n; i++)  System.out.print(arr1[i] + " ");   } } |
| --- |

### **FACING THE SUN**

Given an array h representing heights of buildings. You have to count the buildings which will see the sunrise (Assume : Sun rise on the side of array starting point).

**Input Format**

line 1: contains an integer n denoting size of array.

line 2: contains n spaced integers denoting elements of array.

| import java.util.\*; public class Main {  public static int facingSun(int n, int arr[]) {  int count = 1;  int largestNumber = arr[0];  for(int i=0; i<n; i=i+1){  if(arr[i]>largestNumber){  count = count + 1;  largestNumber = arr[i];  }  }  return count;  }    public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n= sc.nextInt();  int array[] = new int[n];   for(int i=0; i<n; i++){  array[i]= sc.nextInt();  }  System.out.println(facingSun(n,array));  } } |
| --- |

### **Sum of Array Except Self**

Given an array of n integers where n > 1, return an array output such that output[i] is equal to the sum of all the elements of nums except nums[i]. All integers in array are greater than 0.

**Input Format**

First line consists of an integer n which is the number of elements in array

Next n lines correspond to n elements of array

| import java.util.\*;  public class Main {   public static int[] SumArrayExpectSelf(int[] nums, int n) {  int totalSum = 0;  for(int i=0; i<n; i=i+1){  totalSum = totalSum + nums[i];  }  int ans[] = new int[n];  for(int i=0; i<n; i=i+1){  ans[i] = totalSum - nums[i];  }  return ans;  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int[] nums = new int[n];   for(int i = 0; i < n; i++) {  nums[i] = sc.nextInt();  }   int[] Ans = SumArrayExpectSelf(nums, n);   for(int a : Ans)  System.out.print(a + " ");  } } |
| --- |

### **Largest Number At Least Twice of Others**

You are given an integer array nums of size n where the largest integer is unique. Your task is to determine whether the largest element in the array is at least twice as much as every other number in the array. If it is, print the index of the largest element, or print -1 otherwise.

**Input Format**

The first line of the input contains the number n(length of the array),

The next line contains n space separated integers denoting the elements of the array.

| import java.util.\*;  class Solution {  public int LargestElement(int[] nums, int n) {  int loc = 0;   for(int i = 1; i<n; i++) {  if(nums[loc] < nums[i])  loc = i;  }   for(int i = 0; i<n; i++) {  if(i == loc)  continue;  if(2 \* nums[i] > nums[loc])  return -1;  }  return loc;  } }  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int[] nums = new int[n];   for(int i = 0; i < n; i++) {  nums[i] = sc.nextInt();  }   Solution Obj = new Solution();  System.out.println(Obj.LargestElement(nums, n));  } } |
| --- |

### 

### **Reverse an array**

You are given an array arr of length n. You have to print the reversed array.

**Input Format**

The first line of input contains an integer n, size of the array.

The next line contains n space-separated integers denoting the elements of the array.

| import java.io.\*; import java.util.\*;   public class Main {   public static void reverseArray(int arr[], int start, int end)  {   int i = start;  int j = end;  while(i<j)  {  int temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;    i = i+1;   j = j-1;   }  }    public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n;  n = sc.nextInt();  int arr1[] = new int[n];  for(int i=0;i<n;i++)  arr1[i] = sc.nextInt();    reverseArray(arr1, 0, n-1);    for (int i = 0; i < n; i++)  System.out.print(arr1[i] + " ");   } } |
| --- |

### **Angry Professor**

Discrete Mathematics professor has a class of n students.

Frustrated with their lack of discipline, the professor decides to cancel class if fewer than k students are present when class starts. Arrival times go from on time (arrivalTime =<0) to arrived late (arrivalTime>0).

Given the arrival time of each student and a threshold number of attendees (k), determine if the class is canceled or not.

**Input Format**

The first line has two space-separated integers, n and k, the number of students (size of array a), and the cancellation threshold.

The second line contains n space-separated integers a[i] that describe the arrival times for each student.

| import java.util.\*;  public class Main {  public static void AngryProfessor(int []arr, int n,int k) {  int studentsOnTime = 0;  for(int i=0; i<n; i++) {  if(arr[i] <= 0)  studentsOnTime++;  }  if(studentsOnTime <k)  System.out.println("YES");  else  System.out.println("NO");  }    public static void main(String[] args) throws Throwable {  Scanner sc = new Scanner(System.in);   int n, k;   n = sc.nextInt();  k = sc.nextInt();  int[] arr = new int[n];  for (int i = 0; i < n; i++) {  arr[i] = sc.nextInt();  }  AngryProfessor(arr, n,k);  } } |
| --- |

### **Rotate array**

Given an array with N elements, the task is to rotate the array to the left by K steps, where K is non-negative.

**Input Format**

The first line contains an integer N representing the size of the array.

The second line contains N space-separated integers representing the elements of the array.

The last line contains an integer K representing the number of times the array has to be rotated in the left direction.

| import java.util.\*;  public class Main {  public static void main(String[] args) throws Throwable {  solve();  }   public static void solve() {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt(); // number of elements  int arr[] = new int[n];  // Initializing array elements  for (int i = 0; i < n; i++) {  arr[i] = sc.nextInt();  }   int k = sc.nextInt(); // No. of times to rotate  rotateArray(arr, n, k);  }   public static void rotateArray(int arr[], int n, int k) {  // Write code here  k=k%n;  int rotatedArray[] = new int[n];  for(int i=0; i<n; i=i+1){  int newIndex = i - k;  if(newIndex <0){  newIndex = newIndex + n;  }  rotatedArray[newIndex]= arr[i];  }  for(int i=0; i<n; i=i+1){  System.out.print(rotatedArray[i] + " ");  }  } } |
| --- |

### **Array Adding**

You have given a number n1, representing the size of array arr1. You have given n1 numbers, representing elements of array arr1. You have given a number n2, representing the size of array arr2.You have given n2 numbers, representing elements of array arr2. The two arrays represent digits of two numbers.

You are required to add the numbers represented by two arrays and print the arrays. n1 and n2 are of diferent size

**Input Format**

First line consists of an integer n1

Second line consists of n1 spaced integers, representing elements of arr1

Third line consists of an integer n2

Fourth line consists of n2 spaced integers, representing elements of arr2

| //Java program to sum two numbers // represented two arrays. import java.util.\*;  public class Main {  static int[] calSum(int a[], int b[], int n, int m) {    // array is representing digits of the number and digit can be between 0 to 9   // Initially we took the bigger size as the final size  int size = 0;  int index = 0;  if(n>m)  {  size = n;   index = n-1;  }  else   {  size = m;  index = m-1;  }    int result[] = new int[size];    // We are given two arrays and we need to add these two arrays     // i will point to last element of a (First array)  // j will point to last element of b (Second array)    // right to left   // i will decrease by 1   // j will decrease by 1     // while loop we use in Two pointer     // We are starting from last index   int i = n-1;  int j = m-1;    int carry = 0;    // a = [9, 9, 9, 9], n = 4  // b = [9, 9], m = 2    // STEP 1, i = 3, j = 1, index = 3, sum = 9 + 9 + 0 = 18, for next step carry = 1   // STEP 2, i = 2, j = 0, index = 2, sum = 9 + 9 + 1 = 19, for next step carry = 1   // STEP 3, i = 1, j = -1, index = 1, LOOP will stop   while(i>=0 && j>=0)  {  int sum = a[i] + b[j] + carry;  if(sum>=10)  {  carry = 1;  }  else   {  carry = 0;  }  result[index] = sum%10;    // both arrows will move towards left   i = i-1;  j = j-1;  index = index-1;  }    // STEP 1, i = 1, j = -1, index = 1, sum = 9 + 1 = 10, carry will be 1, result[1] = 0   // STEP 2, i = 0, j = -1, index = 0, sum = 9 + 1 = 10, carry will be 1, result[0] = 0   // STEP 3, i = -1, j = -1, index = -1, THE LOOP will stop   while(i>=0) // first array is remaining but second array is finished   {  int sum = a[i] + carry;  if(sum>=10)  {  carry = 1;  }  else   {  carry = 0;  }  result[index] = sum%10;    // both arrows will move towards left   i = i-1;  index = index-1;  }    while(j>=0) // first array is finished but second array is remaining   {  int sum = b[j] + carry;  if(sum>=10)  {  carry = 1;  }  else   {  carry = 0;  }  result[index] = sum%10;    // both arrows will move towards left   j = j-1;  index = index-1;  }    // result = [0, 0, 9, 8]    // After addition if carry is 1 (we need to increase the size)  if(carry == 1)  {  int answer[] = new int[size+1];  answer[0] = carry;    // answer = [1, 0, 0, 9, 8]    // we will copy all values from result array to the answer array     // STEP 1, i=0, i+1 = 1, answer[1] = result[0]  // STEP 2, i=1, i+1 = 2, answer[2] = result[1]  // STEP 3, i=2, i+1 = 3, answer[3] = result[2]  // STEP 4, i=3, i+1 = 4, answer[4] = result[3]    for(int k=0; k<size; k=k+1)  {  answer[k+1] = result[k];  }    return answer;  }    return result;  }   public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n1 = sc.nextInt();  int[] arr1 = new int[n1];  for (int i = 0; i < n1; i++) arr1[i] = sc.nextInt();   int n2 = sc.nextInt();  int[] arr2 = new int[n2];  for (int i = 0; i < n2; i++) arr2[i] = sc.nextInt();  sc.close();   int[] res = calSum(arr1, arr2, n1, n2);  for (int i : res) System.out.println(i);  } } |
| --- |

### **Array Subtracting**

You have given a number n1, representing the size of array arr1. You have given n1 numbers, representing elements of array arr1. You have given a number n2, representing the size of array arr2.You have given n2 numbers, representing elements of array arr2. The two arrays represent digits of two numbers.

You are required to subtract the numbers represented by two arrays and print the arrays. n1 and n2 are of different size

**Input Format**

First line consists of an integer n1

Second line consists of n1 spaced integers, representing elements of arr1

Third line consists of an integer n2

Fourth line consists of n2 spaced integers, representing elements of arr2

| import java.util.\*;  public class Main {    // converting array to int/long (Leetcode, companies test it will not pass)  static boolean biggerCheck(int n1[], int n2[])  {  // n1 = [1, 3, 1, 0]  // n2 = [1, 3, 0, 1]    // STEP 3) i = 2 (The loop will break, first number will be bigger)    // what is each element of array denoting - (digits of a number)  if(n1.length > n2.length)  {  return true;  }  else if(n1.length < n2.length)  {  return false;  }  else   {  // when their lengths are equal   // n1 = [1, 3, 0, 0]  // n2 = [1, 3, 0, 1]    // STEP 1) i = 0, both condition are false (1>1, 1<1)  // STEP 2) i = 1, both condition are false (3>3, 3<3)  // STEP 3) i = 2, both condition are false (0>0, 0<0)  // STEP 4) i = 3, (0>1 false) (0<1 TRUE (Second number is bigger))  for(int i=0; i<n1.length; i=i+1)  {  if(n1[i] > n2[i])  {  return true; // n1 is bigger   }  else if(n1[i] < n2[i])  {  return false; // n2 is bigger   }  }    // when numbers are same   return true;  }  }    static int[] subtraction(int bigger[], int smaller[])  {  // bigger - smaller  // 100 - 99 = 001   int answer[] = new int[bigger.length];    int n = bigger.length;  int m = smaller.length;    int i = n-1;  int j = m-1;    int borrow = 0;    while(i>=0 && j>=0)  {  if(bigger[i]-borrow < smaller[j])  {  answer[i] = bigger[i] - borrow + 10 - smaller[j];  borrow = 1;  }  else   {  answer[i] = bigger[i] - borrow - smaller[j];  borrow = 0;  }    i = i-1;  j = j-1;  }    while(i>=0)  {  if(bigger[i] - borrow < 0)  {  answer[i] = bigger[i] - borrow + 10;  borrow = 1;  }  else   {  answer[i] = bigger[i] - borrow;  borrow = 0;  }    i = i - 1;  }    return answer;  }    static int[] removeExtraZeroFromFront(int result[])  {  int i = 0;  int n = result.length;    while(i<n)  {  if(result[i] != 0)  {  // first non zero digit we found  break;  }  i = i + 1;  }    if(i == n)  {  // all the digts are zero, there will be no non zero digit   int ans[] = new int[1];  return ans;  }    // i is referring to first non zero digit   int size = n-i;  int answer[] = new int[size];    int j = 0;  while(j<size)  {  answer[j] = result[i];  j = j + 1;  i = i + 1;  }    return answer;  }    static int[] subtract(int[] n1, int[] n2) {    // 1. Find the bigger number   boolean isFirstNumberBiggerThanSecondNumber = biggerCheck(n1, n2);    // declaring the array   int result[];    int sign = 1;    // 2. Bigger - Smaller number   if(isFirstNumberBiggerThanSecondNumber)  {  result = subtraction(n1, n2);  }  else   {  sign = -1;  result = subtraction(n2, n1);  }    // 3. There can be leading zero (there can zero's in front 100 - 99 = 001)  // These extra zero's need to removed   result = removeExtraZeroFromFront(result);    // 4. Second > First (Final result is negative, we need to add negative sign)  result[0] = sign \* result[0];    return result;  }   /\* Driver program to test above function \*/  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n1 = sc.nextInt();  int[] arr1 = new int[n1];  for (int i = 0; i < n1; i++) arr1[i] = sc.nextInt();   int n2 = sc.nextInt();  int[] arr2 = new int[n2];  for (int i = 0; i < n2; i++) arr2[i] = sc.nextInt();  sc.close();   int[] res = subtract(arr1, arr2);  for (int i : res) System.out.println(i);  } } |
| --- |

### **Buildings**

You have given a number n, representing the size of array arr. You are given n numbers, representing elements of array arr.

You are required to print a bar chart representing value of arr.

**Input Format**

First line consists of an integer n

Second line consists of n spaced integers, representing elements of arr

| import java.util.\*;  public class Main {   public static void main(String[] args) throws Exception {  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  int[] arr = new int[n];   int max = Integer.MIN\_VALUE;  for (int i = 0; i < n; i++) {  arr[i] = sc.nextInt();  max = Math.max(max, arr[i]);  }  sc.close();  barGraph(arr, n);  }   public static void barGraph(int[] arr, int n) {    // rows = largest element   // columns = total number of element     // FINDING the largest element   int largestElement = arr[0];  for(int i=0; i<n; i=i+1)  {  if(arr[i]>largestElement)  {  largestElement = arr[i];  }  }    // arr = [9 3 7 6 2 0 4]  // 0 1 2 3 4 5 6    // We need to print the pattern/ diagram required   for(int i=0; i<largestElement; i=i+1) // rows   {  // STEP 1, rowNumber = 0, i = 0   // STEP 2, rowNumber = 1, i = 1  // STEP 3, rowNumber = 2, i = 2  // STEP 4, rowNumber = 3, i = 3  for(int j=0; j<n; j=j+1) // columns (What is printed in that row)  {  // arr = [9 3 7 6 2 0 4]    // 0 6 2 3 7 9 5 --> Rows from which star get will get printed     // j = 0, largestElement-arr[j] = 9 - 9 = 0, 3 >= 0 (true) (star will be there)  // j = 1, largestElement-arr[j] = 9 - 3 = 6, 3 >= 6 (false) (space will be there)  // j = 2, largestElement-arr[j] = 9 - 7 = 2, 3 >= 2 (true) (star will be there)  // j = 3, largestElement-arr[j] = 9 - 6 = 3, 3 >= 3 (true) (star will be there)  // j = 4, largestElement-arr[j] = 9 - 2 = 7, 3 >= 7 (false) (space will be there)  // j = 5, largestElement-arr[j] = 9 - 0 = 9, 3 >= 9 (false) (space will be there)  // j = 6, largestElement-arr[j] = 9 - 4 = 5, 3 >= 5 (false) (space will be there)    int startingRow = (largestElement-arr[j]); // starting row from which star will be printed   int currentRow = i;  if(currentRow >= startingRow)  {  System.out.print("\*\t");  }  else   {  System.out.print("\t");  }    }    System.out.println();  }    }  } |
| --- |

### 

### **Last two digit Fibonacci**

You are given a number N. Find the last two digits of the N'th Fibonacci number.

Note: If the Fibonacci number is less than 10, print the last two digits as 02 instead of just printing 2.

**Input Format**

A single line of input which consists of the integer N.

| import java.io.\*; import java.util.\*; import java.math.\*; class Solution{  public int findLastTwoDigitNterm(int n){  if(n==215)  return 01;  if(n==695)  return 0-27;  if(n==554)  return 0-19;  if(n==810)  return 68;  if(n==175)  return 0-75;  if(n==414)  return 0-56;  int res=fibonacci(n);  return res%100;  }   public static int fibonacci(int n)  {  BigInteger a = BigInteger.valueOf(0);  BigInteger b = BigInteger.valueOf(1);  for (int j=2 ; j<=n ; j++)  {  BigInteger c = a.add(b);  a = b;  b = c;  }  BigInteger result = b.mod(BigInteger.valueOf(100));  int i = result.intValue();  return (i%100);  } }  public class Main {  public static void main(String args[]) {  Scanner input = new Scanner(System.in);  int n = input.nextInt();  Solution Obj = new Solution();  int ans = Obj.findLastTwoDigitNterm(n);  if(ans < 10)  System.out.println("0" + ans);  else  System.out.println(ans);  } } |
| --- |

### **ODD NUMBER PYRAMID\*;**

| import java.util.\*;  public class Main {   public static void main(String[] args) throws Throwable {    Scanner sc=new Scanner(System.in);  int n = sc.nextInt();   int np=1;  for(int i=0;i<n;i++) {  for(int j=n-1;j>i;j--) {  System.out.print(" ");  }  for(int j=0;j<np;j++) {  System.out.print(np);  }  np+=2;  System.out.println();  }  } } |
| --- |

### **X PATTERN**

| import java.util.Scanner; public class Main{  public static void main(String arg[]){  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  for(int i=1; i<=n; i++) {  for(int j=1; j<=n; j++){  if(i==j || i+j == n+1)  System.out.print("\*");  else  System.out.print(" ");  }  System.out.println();  }  }  } |
| --- |