IMPORTANT CODES IN JAVA

```
//import java.util.Scanner;
////import java.util.Arrays;
//public class codes100 {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
////        int n = sc.nextInt();
//        positive or negative number
//        if(n>=0) {
//            System.out.println("positive number");
//        }
//        else{
//            System.out.println("negative number");
//        }
//    }
//}
//        even or oddd number
//        if (n % 2 == 0) {
//            System.out.println("even number");
//        } else {
//            System.out.println("odd number");
//        }
//    }
//}
//  sum of fisrt N natural numbers/sum of n natural numbers
//        int sum = 0;
//        for (int i = 1; i <= n; i++) {
//            sum = sum + i;
//        }
//        System.out.println(sum);
//    }
//}
//  using formula
//        int result = n * (n + 1) / 2;
//        System.out.println(result);
//    }
//}
//    using recursion-display the n natural  numbers
//    static void sumOfNatural(int n) {
//        if (n == 0) {
//            return;
//        }
```

```
//       else{
//          sumOfNatural(n-1);
//          System.out.print(n+" ");
//       }
//    }
//    public static void main(String[] args){
//       Scanner sc=new Scanner(System.in);
//       int n=sc.nextInt();
//       sumOfNatural(n);
//    }
//}
//  sum of numbers in a given range
//       int sum = 0, start=5,end=15;
//       for (int i = start; i <= end; i++) {
//          sum = sum + i;
//       }
//       System.out.println(sum);
//    }
//}
//  greatest of two numbers
//       int a = sc.nextInt();
//       int b = sc.nextInt();
//       if (a > b) {
//          System.out.println(a + " is the greatest number");
//       } else {
//          System.out.println(b + " is the greatest number");
//       }
//    }
//}
//  greatest of three numbers
//       int c = sc.nextInt();
//       if (a > b && a > c) {
//          System.out.println(a + " is the greatest number");
//       } else if (b > a && b > c) {
//          System.out.println(b + " is the greatest number");
//       } else {
//          System.out.println(c + " is the greatest number");
//       }
//    }
//}
//  leap year or not
//       int n = sc.nextInt();
//       if (n % 4 == 0 || n % 400 == 0 && n % 100 != 0) {
//          System.out.println(n + " is the leap year");
```

```
//      } else {
//          System.out.println(n + " is not a leap year");
//      }
//  }
//}
//  prime number or not
//      int n = sc.nextInt();
//      if (n <= 0) {
//          System.out.println(n + " is not a prime number");
//      } else {
//          int flag = 0;
//          for (int i = 2; i < n; i++) {
//              if (n % i == 0) {
//                  flag = 1;
//                  break;
//              }
//          }
//          if (flag == 0) {
//              System.out.println("prime number");
//          } else {
//              System.out.println("not a prime number");
//          }
//      }
//  }
//}
//  prime number within a given range
//      int start = sc.nextInt(), end = sc.nextInt();
//      for (int i = start; i <= end; i++) {
//          if (i > 1) {
//              int flag = 0;
//              for (int j = 2; j <= i / 2; ++j) {
//                  if (i % j == 0) {
//                      flag = 1;
//                      break;
//                  }
//              }
//              if (flag == 0) {
//                  System.out.print(i + " ");
//              }
//          }
//      }
//  }
//}
//  reverse of a number
```

```
//      int n = sc.nextInt();
//      int reverse = 0;
//      while (n != 0) {
//         int digit = n % 10;
//         reverse = reverse * 10 + digit;
//         n = n / 10;
//      }
//      System.out.println(reverse);
//   }
//}
//  sum of digits of a number
//      int sum = 0;
//      while (n > 0) {
//         int digit = n % 10;
//         sum = sum + digit;
//         n = n / 10;
//      }
//      System.out.println(sum);
//   }
//}
// Palindrome number
//      int sum = 0;
//      int temp = n;
//      while (n != 0) {
//         int digit = n % 10;
//         sum = sum * 10 + digit;
//         n = n / 10;
//      }
//      if (sum == temp) {
//         System.out.println(temp + " is a palindrome number");
//      } else {
//         System.out.println(temp + " is not a palindrome number");
//      }
//   }
//}
//   Armstrong number
//      int sum = 0;
//      int temp = n;
//      while (n != 0) {
//         int digit = n % 10;
//         sum = sum + digit*digit*digit;
//         n = n / 10;
//      }
//      if (sum == temp) {
```

```
//            System.out.println(temp + " is a armstrong number");
//        } else {
//            System.out.println(temp + " is not a armstrong number");
//        }
//    }
//}
//  Armstrong numbers in the given range
//        int start = sc.nextInt(), end = sc.nextInt();
//        for (int i = start; i < end; i++) {
//            int sum = 0;
//            int temp = i;
//            while (temp != 0) {
//                int digit = temp % 10;
//                sum = sum + (digit * digit * digit);
//                temp = temp  / 10;
//            }
//            if (sum == i) {
//                System.out.print(i + " ");
//            }
//        }
//    }
//}
//  fibonacci series upto nth term or nth fibonacci number
//        int n = sc.nextInt();
//        int n1 = 0, n2 = 1, n3;
//        System.out.print(n1+" "+n2+" ");
//        for (int i = 0; i <= n; i++) {
//            n3 = n1 + n2;
//            System.out.print(n3 + " ");
//            n1 = n2;
//            n2 = n3;
//        }
//    }
//}
//  Factorial of a number
//        int n = sc.nextInt();
//        int fact = 1;
//        for (int i = 1; i <= n; i++) {
//            fact = fact * i;
//        }
//        System.out.println(fact + " ");
//    }
//}
//    factorial of a number using the recursion
```

```
//    static int factorial(int n) {
//        if (n == 0) {
//            return 1;
//        } else {
//            return n * factorial(n - 1);
//        }
//    }
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int num = sc.nextInt();
//        System.out.println(factorial(num));
//    }
//}
//  power of a number
//        double a = sc.nextInt();
//        double b = sc.nextInt();
//        double result = Math.pow(a, b);
//        System.out.println(result);
//    }
//}
//  factor of a number
//        int n = sc.nextInt();
//        for (int i = 1; i <= n; i++) {
//            if (n % i == 0) {
//                System.out.print(i + " ");
//            }
//        }
//    }
//}
//  prime factors of a number
//        for (int i = 2; i < n; i++) {
//            if (n % i == 0) {
//                System.out.print(i + " ");
//                n=n/i;
//            }
//        }
//        if(n>2){
//            System.out.print(n+" ");
//        }
//    }
//}
//   Strong number or not a strong number
//        int temp = n;
//        int sum = 0;
```

```
//       while (temp!= 0) {
//          int digit = temp% 10;
//          int fact = 1;
//          for (int i = 2; i <= digit; i++){
//             fact = fact * i;
//       }
//       sum = sum + fact;
//       temp= temp / 10;
//      }
//       if(sum==n) {
//          System.out.println(n+ " is a strong number");
//       } else {
//          System.out.println(n+ " is not an strong number");
//       }
////    }
//}
//  perfect number or not a perfect number
//       int temp = n;
//       int sum = 0;
//       for (int i = 1; i < n; i++) {
//          if (n % i == 0) {
//             sum = sum + i;
//          }
//       }
//       if (sum == temp) {
//          System.out.println(temp + " is a perfect number");
//       } else {
//          System.out.println(temp + " is not a perfect number");
//       }
//    }
//}
//  perfect square number
//       double n = sc.nextDouble();
//       double res = (Math.sqrt(n));
//       System.out.println((int) res);
//       if (n == (res * res)) {
//          System.out.println(n + " is a perfect square number");
//       } else {
//          System.out.println(n + " is not a perfect square number");
//       }
//    }
//}
//  Automorphic number
//       int n = sc.nextInt();
```

```
//        int res =n*n;
//        int digit = res % 10;
//        if (n == digit) {
//            System.out.println(n + " is a automorphic number");
//        } else {
//            System.out.println(n + " is not a automorphic number");
//        }
//    }
//}
//  Harshad number
//        int n = sc.nextInt();
//        int temp=n;
//        int sum = 0;
//        while (n != 0) {
//            int digit = n % 10;
//            sum = sum + digit;
//            n=n/10;
//        }
//        if (temp % sum == 0) {
//            System.out.println(temp + " is a harshad number");
//        } else {
//            System.out.println(temp + " is not a harshad number");
//        }
//    }
//}
//  Abundant number
//        int n = sc.nextInt();
//        int sum = 0;
//        for (int i = 1; i < n; i++) {
//            if (n % i == 0) {
//                sum = sum + i;
//            }
//        }
//        if (sum > n) {
//            System.out.println(n + " is a abundant number");
//        } else {
//            System.out.println(n + " is not a abundant number");
//        }
//    }
//}
//   Friendly pair
//        int n1 = sc.nextInt();
//        int n2 = sc.nextInt();
//        int sum1 = 0;
```

```
//        for (int i = 1; i < n1; i++) {
//            if (n1 % i == 0) {
//                sum1 = sum1 + i;
//            }
//        }
//        int sum2 = 0;
//        for (int i = 1; i < n2; i++) {
//            if (n2 % i == 0) {
//                sum2 = sum2 + i;
//            }
//        }
//        int res1 = sum1 / n1;
//        int res2 = sum2 / n2;
//        if (res1 == res2) {
//            System.out.println(n1 + " " + n2 + " are friendly pair");
//        } else {
//            System.out.println(n1 + " " + n2 + " are not friendly pair");
//        }
//    }
//}
//  HCF of two numbers/or also called as the gcd(greatest common divisior)
//        int num1 = 36, num2 = 24, hcf = 1;
//        for (int i = 1; i <= num1 || i <= num2; i++) {
//            if (num1 % i == 0 && num2 % i == 0) {
//                hcf = i;
//            }
//        }
//        System.out.println(hcf);
//    }
//}
//        lcm of two numbers
//        int num1 = 36, num2 = 24, hcf = 1;
//        for (int i = 1; i <= num1 || i <= num2; i++) {
//            if (num1 % i == 0 && num2 % i == 0) {
//                hcf = i;
//            }
//        }
//        int lcm=(num1*num2)/hcf;
//        System.out.println("lcm of two numbers is"+lcm);
//    }
//}
//  binary to decimal conversion
//        int binary = sc.nextInt();
//        int decimal = 0;
```

```
//      int n = 0;
//      while (binary != 0) {
//         int digit = binary % 10;
//         decimal += digit * Math.pow(2,n);
//         binary = binary / 10;
//         n++;
//      }
//      System.out.println(decimal);
//   }
//}
//  hexadecimal to decimal number
//      int hexadecimal = sc.nextInt();
//      int decimal = 0;
//      int n = 0;
//      while (hexadecimal != 0) {
//         int digit = hexadecimal % 10;
//         decimal += digit * Math.pow(8,n);
//         hexadecimal = hexadecimal / 10;
//         n++;
//      }
//      System.out.println(decimal);
//   }
//}
//  Hexadecimal to decimal conversion
//      int hexadecimal = sc.nextInt();
//      int decimal = 0;
//      int n = 0;
//      while (hexadecimal != 0) {
//         int digit = hexadecimal % 10;
//         decimal += digit * Math.pow(16,n);
//         hexadecimal = hexadecimal / 10;
//         n++;
//      }
//      System.out.println(decimal);
//   }
//}
//  decimal to binary conversion
//      int n = sc.nextInt();
//      String result = "";
//      while (n >=1) {
//         int digit = n % 2;
//         n = n / 2;
//         result = digit+result;
//      }
```

```
//       System.out.println(result);
//    }
//}
//   decimal to octal conversion
//       int n = sc.nextInt();
//       String result = "";
//       while (n >= 1) {
//          int digit = n % 8;
//          n = n / 8;
//          result = digit + result;
//       }
//       System.out.println(result);
//    }
//}
//  decimal to hexadecimal
//       int n = sc.nextInt();
//       String result = "";
//       while (n >= 1) {
//          int digit = n % 16;
//          n = n / 16;
//          result = digit + result;
//       }
//       System.out.println(result);
//    }
//}
//  binary to octal
//    steps:binary to decimal and decimal to octal
//       int binary = sc.nextInt();
//       int n = 0;
//       int decimal = 0;
//       while (binary > 0) {
//          int digit = binary % 10;
//          decimal += digit * Math.pow(2, n);
//          binary = binary / 10;
//          n++;
//       }
//       System.out.println(decimal);
//       int n = sc.nextInt();
//       String result = "";
//       while (n >= 1) {
//          int digit = n % 8;
//          n = n / 8;
//          result = digit + result;
//       }
```

```
//       System.out.println(result);
//    }
//}
//  octal to binary conversion=octal to decimal and then decimal to binary
//       int  octal= sc.nextInt();
//       int n = 0;
//       int decimal = 0;
//       while (octal > 0) {
//          int digit = octal % 10;
//          decimal += digit * Math.pow(8, n);
//          octal = octal / 10;
//          n++;
//       }
//       System.out.println(decimal);
//       int n = sc.nextInt();
//       String result = "";
//       while (n >= 1) {
//          int digit = n % 16;
//          n = n / 16;
//          result = digit + result;
//       }
//       System.out.println(result);
//    }
//}
//  points lies in which quadrant
//       int x = sc.nextInt();
//       int y = sc.nextInt();
//       if (x > 0 && y > 0) {
//          System.out.println(x + " , " + y + " lies on the quadrant one");
//       } else if (x < 0 && y > 0) {
//          System.out.println(x + " , " + y + " lies on the quadrant two");
//       } else if (x < 0 && y < 0) {
//          System.out.println(x + " , " + y + " lies on the quadrant three");
//       } else {
//          System.out.println(x + " , " + y + " lies on the quadrant four");//x>0 and y<0
//       }
//    }
//}
//  permutation like n people occupy r places in a class in java
//       int n = sc.nextInt();
//       int r = sc.nextInt();
//       int fact1 = 1;
//       for (int i = 2; i <= n; i++) {
//          fact1 = fact1 * i;
```

```
//      }
//      int fact2 = 1;
//      for (int i = 2; i <= (n - r); i++) {
//         fact2 = fact2 * i;
//      }
//      int result = fact1 / fact2;
//      System.out.println(result);
//   }
//}
//   Maximum number of handshakes
//   formual is there to find the number of handshakes
//      int N = sc.nextInt();
//      int handshakes = N * (N - 1) / 2;
//      System.out.println(handshakes);
//   }
//}
//  addition of two fractions
//      int num1 = sc.nextInt();
//      int den1 = sc.nextInt();
//      int num2 = sc.nextInt();
//      int den2 = sc.nextInt();
//      int numerator3 = ((num1 * den2) + (num2 * den1));
//      int denominator3 = den1 * den2;
//      System.out.println(numerator3 + "/" + denominator3);
//   }
//}
//   replace 0's with 1's and viceversa
//      int n = sc.nextInt();
//      int result=0;
//      int place=1;
//      while (n > 0) {
//         int digit = n % 10;
//         if (digit == 0) {
//            digit = 1;
//         }
//         result=result+(digit*place);
//         place=place*10;
//         n = n / 10;
//      }
//      System.out.println(result);
//   }
//}
//  program to express the number as sum of two prime numbers
//   static int isPrime(int n) {
```

```
//      int flag = 0;
//      for (int i = 2; i < n; i++) {
//         if (n % i == 0) {
//            flag = 1;
//            break;
//         }
//      }
//      return flag;
//   }
//   public static void main(String[] args) {
//      Scanner sc = new Scanner(System.in);
//      int n = sc.nextInt();
//      int flag = 0;
//      for (int i = 2; i <= n / 2; i++) {
//         if (isPrime(i)==0) {
//            if (isPrime(n - i)==0) {
//               System.out.println(n + "=" + i +" +"+ (n - i));
//               flag = 1;
//            }
//         }
//      }
//      if (flag == 0) {
//         System.out.println(n + " cannot be represnted in the form of sum of two prime
numbers");
//      }
//   }
//}
//  area of a circle
//      double r = sc.nextDouble();
//      double area = 3.14* r * r;
//      System.out.println("area of circle is " + Math.round(area));
//   }
//}
//   calculate number of digits in an integer
//      int n = sc.nextInt();
//      int count = 0;
//      while (n != 0) {
//         int digit = n % 10;
//         count++;
//         n = n / 10;
//      }
//      System.out.println(count);
//   }
//}
```

```
//    convert numbers/digits to words(in the rane of (0-99)
//       int n=sc.nextInt(); //for example=12
//       int n1=n,n2=n;//12
//       int a=n1%10,b=n2/10;//a=2,b=1
//       String[] onesplace=new
String[]{"zero","one","two","three","four","five","six","seven","eight","nine"};
//       String[] tensplace=new
String[]{"","ten","eleven","twelve","thirteen","fourteen","fifteen","sixteen","seventeen","ei
ghteen","ninteen"};
//       String[] hundredsplace=new String[]{" ","
","twenty","thirty","fourty","fifty","sixty","seventy","eighty","ninety"};
//        if(b==1) { //b==1 true
//           System.out.println(tensplace[a+1]);//2+1=3
//        }
//        else if(a==0) { //not excecuted
//           System.out.println(hundredsplace[b]);
//        }
//        else {
//           System.out.println(hundredsplace[b] + "-" + onesplace[a]);
//        }
//    }
//}
//    print number of days in a month
//       int month = sc.nextInt();
//       int year = sc.nextInt();
//       if (month==2&&year % 4 == 0 || year % 400 == 0 && year % 100 != 0) {
//           System.out.println("29 days are there in that month");
//       }
//       else if(month==2) {
//           System.out.println("28 days in a month");
//       }
//       else if (month == 1 || month == 3 || month == 5 || month == 7 || month == 9 ||
month == 11) {
//           System.out.println("31 days are there in given month");
//       } else {
//           System.out.println("30 days are there in a given month");
//       }
//    }
//}
//    counting the frequency of a particular number in the given number
//       int n = sc.nextInt();
//       int count = 0;
//       int d = 9;
//       while (n != 0) {
```

```
//          int digit = n % 10;
//          if (digit == d) {
//              count++;
//          }
//          n = n / 10;
//      }
//      System.out.println(count);
//   }
//}
//  counting the frequency of the digits in the given number
//      int n = sc.nextInt();
//      int count[] = new int[10];
//      while (n != 0) {
//          int digit = n % 10;
//          count[digit]++;
//          n = n / 10;
//      }
//      for (int i = 0; i < 10; i++) {
//          if (count[i] > 0) {
//              System.out.println("digit "+i+"-->"+count[i]+" times");
//          }
//      }
//   }
//}
//  count the integers that has exactly x divisiors
//      int n = sc.nextInt();
//      int x = sc.nextInt();
//      int count = 0;
//      for (int i = 1; i <= n; i++) { //1 to 8
//          int count_digits = 0;
//          for (int j = 1; j <= i; j++) { // 1 to 8
//              if (i % j == 0) {
//                  count_digits++;
//              }
//          }
//          if (count_digits == x) {
//              count++;
//          }
//      }
//      System.out.println(count);
//   }
//}
//      double a = sc.nextDouble();
//      double b = sc.nextDouble();
```

```
//      double c = sc.nextDouble();
//      double determinant = (b * b) - 4 * a * c;
//      if (determinant > 0) {
//         double root1 = -b + Math.sqrt(determinant) / (2 * a);
//         double root2 = -b - Math.sqrt(determinant) / (2 * a);
//         System.out.println("The roots are real and distinct");
//         System.out.println(root1 + "," + root2 + " are the roots");
//      } else if (determinant == 0) {
//         double root1 = -b / (2 * a);
//         System.out.println(root1 + " it has only one real root");
//      } else {
//         double root1 = -b / (2 * a);
//         double root2 =  Math.sqrt(-determinant) / (2 * a);
//         System.out.println("roots are imaginary");
//         System.out.println(Math.round(root1) + "+" + "i" + Math.round(root2));
//      }
//   }
//}
// power of a number without using any inbuilt functions/means using recursion we solve
the problem
//      double a = sc.nextDouble();
//      double b = sc.nextDouble();
//      System.out.println(powerofnumber(a,b));
//   }
//   static double powerofnumber(double a,double b) {
//      if (b == 0) {
//         return 1;
//      } else {
//         return a * powerofnumber(a, b - 1);
//      }
//   }
//}
//  largest element in the array-method 1
//      int n = sc.nextInt();
//      int a[] = new int[n];
//      for (int i = 0; i < n; i++) {
//         a[i] = sc.nextInt();
//      }
//      Arrays.sort(a);
//      System.out.println(Arrays.toString(a));
//      int largest = a[n - 1];
//      System.out.println(largest);
//   }
//}
```

```
//       int ans=a[0];
//       for(int i=1;i<n;i++) {
//          if (ans < a[i]) {
//             ans=a[i];
//          }
//       }
//       System.out.print(ans+" ");
//    }
//}
//   smallest element in the array
//       int ans=a[0];
//       for(int i=1;i<n;i++) {
//          if (ans > a[i]) {
//             ans=a[i];
//          }
//       }
//       System.out.print(ans+" ");
//    }
////}
//       Arrays.sort(a);
//       System.out.println(Arrays.toString(a));
//       int smallest = a[0];
//       System.out.println(smallest);
//    }
//}
//  calculate the length of the string without using length() function
//  import java.util.Scanner;
//  public class importantcoding {
//      public static void main(String[] args) {
//         Scanner sc = new Scanner(System.in);
//         String s = sc.nextLine();
//         int count = 0;
//         for (int i = 0; i < s.length(); i++) {
//            count++;
//         }
//         System.out.println(count);
//      }
//  }
//palindrome strings
//import java.util.Scanner;
//public class importantcoding {
//    public static void main(String[] args) {
//       Scanner sc = new Scanner(System.in);
//       String s = sc.nextLine();
```

```
//        char[] arr = s.toCharArray();
//        System.out.println(palindrome(arr));
//    }
//
//    static boolean palindrome(char[] arr) {
//        int left = 0;
//        int right = arr.length - 1;
//        while (left < right) {
//            if (arr[left] != arr[right]) {
//                return false;
//            } else {
//                left++;
//                right--;
//            }
//        }
//        return true;
//    }
//}
//import java.util.Scanner;
//public class codes100{
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int n = sc.nextInt();
//        int marks[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            marks[i] = sc.nextInt();
//        }
//        int count = 0;
//        int ans = marks[0];
//        for (int i = 1; i < n; i++) {
//            if (ans == marks[i]) {
//                count++;
//            }
//        }
//        if (count > 0) {
//            count++;
//        }
//        System.out.println(count);
//    }
//}
// arrays programs-largest element in the array
//import java.util.Scanner;
//import java.util.Arrays;
//public class codes100 {
```

```
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int n = sc.nextInt();
//        int a[] = new int[n];
//        for (int i = 0; i < n; i++) {
//            a[i] = sc.nextInt();
//        }
//        Arrays.sort(a);
//        System.out.println(Arrays.toString(a));
//        int result = a[n - 1];
//        System.out.println(result);
//    }
//}
//        int ans=a[0];
//        for(int i=1;i<n;i++){
//            if(ans<a[i]){
//                ans=a[i];
//            }
//        }
//        System.out.println(ans);
//    }
//}
//arrays-samllest element in the array
//        Arrays.sort(a);
//        System.out.println(Arrays.toString(a));
//        int result = a[0];
//        System.out.println(result);
//    }
//}
//        int ans = a[0];
//        for (int i = 1; i < n; i++) {
//            if (ans > a[i]) {
//                ans = a[i];
//            }
//        }
//        System.out.println(ans);
//    }
//}
//  combinationn of smallest and largest element in the array
//        int ans=a[0];
//        for(int i=1;i<n;i++){
//            if(ans<a[i]){
//                ans=a[i];
//            }
```

```
//      }
//      System.out.println(ans);
//      for (int i = 1; i < n; i++) {
//          if (ans > a[i]) {
//              ans = a[i];
//          }
//      }
//      System.out.println(ans);
//   }
//}
// Second largest element in the array
//      Arrays.sort(a);
//      System.out.println(Arrays.toString(a));
//      int result = a[1];
//      System.out.println(result);
//   }
//}
//      Arrays.sort(a);
//      System.out.println(Arrays.toString(a));
//      int result = a[n-2];
//      System.out.println(result);
//   }
//}
// sum of the elements in the array
//      int sum = 0;
//      for (int i = 0; i < n; i++) {
//          sum = sum + a[i];
//      }
//      System.out.println(sum);
//   }
//}
// Reverse of an array-method 1
//      int start = 0;
//      int end = a.length - 1;
//      while (start < end) {
//          int temp = a[start];
//          a[start] = a[end];
//          a[end] = temp;
//          start++;
//          end--;
//      }
//      for(int i=0;i<n;i++)
//      System.out.print(a[i]+" ");
//   }
```

```
//}
//      method-2 for reverse of an array
//      for (int i = a.length - 1; i >= 0; i--) {
//         System.out.print(a[i] + " ");
//      }
//   }
//}
//  sort the first half in ascending order and second half in descending order
//      Arrays.sort(a);
//      System.out.println(Arrays.toString(a));
//      for (int i = 0; i < n / 2; i++) {
//         System.out.print(a[i]+" ");
//      }
//         for (int j = n-1; j >=n/2; j--) {
//         System.out.print(a[j]+" ");
//      }
//   }
//}
//sort the array:method-1 using the bubble sort or any sorting algorithms
//import java.util.Arrays;
//import java.util.*;
//public class codes100 {
//   public static void main(String[] args) {
//      int[] a = {5, 4, 1,3,2};
//      bubblesort(a);
//      System.out.println(Arrays.toString(a));
//   }
//   static void bubblesort(int[] a) {
//      for (int i = 0; i < a.length; i++) {
//         int swap = 0;
//         for (int j = 1; j < a.length - i; j++) {
//            if (a[j-1] > a[j]) {
//               int temp = a[j-1];
//               a[j-1] = a[j];
//               a[j] = temp;
//               swap = 1;
//            }
//         }
//         if (swap == 0) {
//            break;
//         }
//      }
//   }
//}
```

```
//  finding frequency of the number in the array
//       for (int i = 0; i < n; i++) {
//          int flag = 0;
//          int count = 0;
//          for (int j = i + 1; j < n; j++) {
//             if (a[i] == a[j]) {
//                flag = 1;
//                break;
//             }
//          }
//          if (flag == 1)
//             continue;
//          for (int j = 0; j <= i; j++) {
//             if (a[i] == a[j])
//                count++;
//          }
//          System.out.println(a[i] + " : " + count);
//       }
//    }
//}
// anagrams
//import java.util.Arrays;
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//       Scanner sc = new Scanner(System.in);
//       String str1 = sc.nextLine(); // eat
//       String str2 = sc.nextLine(); // ate
//       if (str1.length() == str2.length()) {
//          char char1[] = str1.toCharArray();
//          char char2[] = str2.toCharArray();
//          Arrays.sort(char1);
//          Arrays.sort(char2);
//          boolean result = Arrays.equals(char1, char2);
//          if (result) {
//             System.out.println("Anagrams");
//          } else {
//             System.out.println("Not Anagrams");
//          }
//       } else {
//          System.out.println("Not Anagram");
//
//       }
//    }
```

```
//}
//java program for the checking whether the given character is uppercase or lowercase
//import java.util.Scanner;
//public class codes100 {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        char ch = sc.next().charAt(0);
//        if (ch >='A' && ch <= 'Z') {
//            System.out.println(1);
//        } else if (ch >= 'a' && ch <= 'z') {
//            System.out.println(0);
//        } else {
//            System.out.println(-1);
//        }
//    }
//}
//import java.util.*;
//import java.util.Arrays;
//public class codes100 {
//    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();
//        }
//        Arrays.sort(arr);
//        System.out.println(Arrays.toString(arr));
//        int ans = arr[0];
//        int result = arr[n - 1] + arr[n - 2];
//        System.out.println(result);
//    }
//}
//import java.util.Scanner;
//public class codes100 {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int n = sc.nextInt();
//        int count = 0;
//        int number=sc.nextInt();
//        while(n!=0){
//            int digit=n%10;
//            if(digit==number){
//                count++;
```

```
//          }
//          n=n/10;
//      }
//      System.out.println(count);
//   }
//}
// calculate the number of vowels and consonants in the given string
//import java.util.*;
//public class codes100 {
//   public static void main(String[] args) {
//      Scanner sc = new Scanner(System.in);
//      String str = sc.nextLine();
//      int vowelcount = 0, consonantcount = 0,whitespaces=0;
//      for (int i = 0; i < str.length(); i++) {
//          char ch=str.charAt(i);
//          if (ch== 'a'|| ch == 'e'||ch  == 'i' || ch == 'o' || ch == 'u') {
//              vowelcount++;
//          }
//          else if(ch==' '){
//              whitespaces++;
//          }
//          else {
//              consonantcount++;
//          }
//      }
//      System.out.println("number of vowels "+vowelcount);
//      System.out.println("number of consonants "+consonantcount);
//      System.out.println("number of white spaces "+whitespaces);
//   }
//}
//find the ascii value of the character-using type casting we can achieve the following code
//import java.util.Scanner;
//import java.util.*;
//public class codes100 {
//   public static void main(String[] args) {
//      Scanner sc = new Scanner(System.in);
//      char ch = sc.next().charAt(0);
//      int result=ch;
//      System.out.println(result); //ascii value of lowercase letter starts from 97 and
uppercase starts from 65
//   }
//}
//remove the vowels from the given string
//import java.util.Scanner;
```

```
//import java.util.*;
//public class codes100{
//    public static void main(String[] args) {
//        Scanner sc=new Scanner(System.in);
//        String str=sc.nextLine();
//        String result=str.replaceAll("[aeiouAEIOU]","");
//        System.out.println(result);
//    }
//}
//import java.util.Scanner;
//import java.util.*;
//public class codes100{
//    public static void main(String[] args) {
//        Scanner sc=new Scanner(System.in);
//        String str=sc.nextLine();
//        String result=str.replaceAll(" ","");
//        System.out.println(result);
//    }
//}
//import java.util.Scanner;
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        StringBuffer string = new StringBuffer();
//        String str1 = sc.nextLine();
//        str1.toLowerCase();
//        for (int i = 0; i < str1.length(); i++) {
//            if (str1.charAt(i) >= 'a' || str1.charAt(i) <= 'z') {
//                string.append(str1.charAt(i));
//            }
//        }
//    }
//}
//sum of the unique elements in the array
//import java.util.*;
//public class codes100{
//    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<arr.length;i++){
//            arr[i]=sc.nextInt();
//        }
```

```
//      int sum=0;
//      for( int i=0;i<arr.length;i++){
//          int j;
//          for(j=0;j<i;j++){
//              if(arr[i]==arr[j]){
//                  break;
//              }
//          }
//          if(i==j){
//              sum+=arr[i];
//          }
//      }
//      System.out.println(sum);
//  }
//}
//print the sum of prime index elements in the array
//import java.util.*;
//public class codes100 {
//  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 < arr.length; i++) {
//          arr[i] = sc.nextInt();
//      }
//      int sum = 0;
//      for (int i = 2; i < arr.length; i++) {
//          int flag = 0;
//          for (int j = 2; j <=Math.sqrt(i); j++) {
//              if (i % j == 0) {
//                  flag = 1;
//                  break;
//              }
//          }
//          if (flag == 0) {
//              sum = sum + arr[i];
//          }
//      }
//      System.out.println(sum);
//  }
//}
//import java.util.*;
//public class codes100 {
//  public static void main(String[] args) {
```

```
//        Scanner sc = new Scanner(System.in);
//        float f = sc.nextFloat();
//        float result = (int) (3.14 * f * f);
//        System.out.println(result);
//    }
//}
//counting the unique elments/distinct elements in the array
//import java.util.*;
//public class codes100{
//    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<arr.length;i++) {
//            arr[i] = sc.nextInt();
//        }
//        int count=0;
//        for(int i=0;i<arr.length;i++){
//            int j;
//            for( j=0;j<i;j++){
//                if(arr[i]==arr[j]){
//                    break;
//                }
//            }
//        if(i==j){
//            count++;
//        }
//        }
//        System.out.println(count);
//    }
//}
//program for counting the number of odd and even elements in the array
//import java.util.*;
//public class codes100 {
//    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 < arr.length; i++) {
//            arr[i] = sc.nextInt();
//        }
//        int oddcount = 0, evencount = 0;
//        for (int i = 0; i < arr.length; i++) {
//            if (arr[i] % 2 == 0) {
```

```
//             oddcount++;
//         } else {
//             evencount++;
//         }
//     }
//     System.out.println(oddcount);
//     System.out.println(evencount);
//  }
//}
//****two pointer techinque****for removing the duplicate elements from the sorted array
//import java.util.*;
//public class codes100 {
//  public static void main(String[] args) {
//     int[] arr = {1, 1, 2, 2, 3, 4, 5,5,6, 6};
//     int newlength=removeduplicate(arr);
//     for (int i = 0; i < newlength; i++) {
//         System.out.print(arr[i]+" ");
//     }
//  }
//  static int removeduplicate(int[] arr) {
//     if(arr.length==0){
//         return -1;
//     }
//     int rd = 0;
//     for (int i = 1; i < arr.length; i++) {
//         if (arr[rd] != arr[i]) {
//             rd++;
//             arr[rd] = arr[i];
//         }
//     }
//     return rd+ 1;
//  }
//}
//swapping of two numbers using different methods
//import java.util.Scanner;
//public class codes100 {
//  public static void main(String[] args) {
//     Scanner sc = new Scanner(System.in);
//     int a = sc.nextInt();
//     int b = sc.nextInt();
////     first method-using temporary variable
//     int temp = a;
//     a = b;
//     b = temp;
```

```
////       second method using arithmetic operators
//       a=a+b;
//       b=a-b;
//       a=a-b;
////      third method also using arithemtic operators
//       a=a*b;
//       b=a/b;
//       a=a/b;
////       fourth method-using bitwise operators
//       a=a^b;
//       b=a^b;
//       a=a^b;
//       System.out.println("after swapping a is " + a);
//       System.out.println("after swapping b is " + b);
//   }
//}
//remove the duplicates from the array using the hashset
//import java.util.*;
//public class codes100{
//    public static void main(String[] args) {
//       Scanner sc = new Scanner(System.in);
//       int n = sc.nextInt();
//       int[] a = new int[n];
//       Set hash = new HashSet<>();
//       for (int i = 0; i < a.length; i++) {
//          a[i] = sc.nextInt();
//          hash.add(a[i]);
//       }
//       System.out.println(hash);
//    }
//}
//fibonacci number using the recursion in java
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//       Scanner sc = new Scanner(System.in);
//       int n = sc.nextInt();
//       System.out.println(fibonacci(n));
//    }
//    static int fibonacci(int n) {
//       if (n == 0) {
//          return 0;
//       }
//       else if(n==1){
```

```
//          return 1;
//      }
//      else {
//          return fibonacci(n-1) + fibonacci(n - 2);
//      }
//   }
//}
//sum of the natural numbers uisng the recursion
//import java.util.*;
//public class codes100 {
//   public static void main(String[] args) {
//      Scanner sc = new Scanner(System.in);
//      int n = sc.nextInt();
//      System.out.println(sumofnnatural(n));
//   }
//   static int sumofnnatural(int n) {
//      if (n == 0) {
//          return 0;
//      } else if (n == 1) {
//          return 1;
//      } else {
//          return n + sumofnnatural(n - 1);
//      }
//   }
//}
//display the natural numbers using the recursion
//import java.util.*;
//public class codes100 {
//   public static void main(String[] args) {
//      Scanner sc = new Scanner(System.in);
//      int n = sc.nextInt();
//     sumofnnatural(n);
//   }
//   static void sumofnnatural(int n){
////      if (n >= 1) {
////          System.out.print(n + " ");
////          sumofnnatural(n - 1); //it will print the natural numbers in the reverse
//      if (n >= 1) {
//          sumofnnatural(n - 1);
//          System.out.print(n + " ");
//      }
//   }
//}
//finding the missing number using java
```

```
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int size = sc.nextInt() ;
//        int[] a = new int[size];
//        for (int i = 0; i < a.length; i++) {
//            a[i] = sc.nextInt();
//        }
//        int n=a.length+1;
//        int expectedsum = (n * (n + 1)) / 2;
//        int actualsum = 0;
//        for (int i = 0; i < size; i++) {
//            actualsum += a[i];
//        }
//        int missingnumber = expectedsum - actualsum;
//        System.out.println(missingnumber);
//    }
//}
// number of stairs he can climb to reach the final stair.
//import java.util.Scanner;
//public class codes100{
//    public static void main(String[] args) {
//        Scanner scanner = new Scanner(System.in);
//        int T = scanner.nextInt();
//        for (int i = 0; i < T; i++) {
//            int X = scanner.nextInt();
//            int Y = scanner.nextInt();
//            int ySteps = X / Y;
//            int remainingSteps = X % Y;
//            int totalMoves = ySteps + remainingSteps;
//            System.out.println(totalMoves);
//        }
//        scanner.close();
//    }
//}
//generae the paranthesis using recursion-we can use any brackets like (),{}.[] any one of
them we can use
//import java.util.Scanner;
//public class codes100{
//    public static void main(String[] args) {
//        Scanner scanner = new Scanner(System.in);
//        int n = scanner.nextInt();
//        char[] a = new char[n * 2];
```

```
//     generateparenthesis(a,n,0,0,0);
//   }
//   static void generateparenthesis(char[] a,int n,int i,int o ,int c){
//     if(i==a.length) {
//        System.out.println(a);
//        return;
//     }
//     if(o<n) {
//        a[i] = '(';
//        generateparenthesis(a, n, i + 1, o + 1, c);
//     }
//     if(c<o) {
//        a[i] = ')';
//        generateparenthesis(a, n, i + 1, o, c + 1);
//     }
//   }
//}
//find the subsequences of the given string using recursion
//import java.util.Scanner;
//import java.util.ArrayList;
//public class codes100{
//   public static void main(String[] args) {
//     Scanner scanner = new Scanner(System.in);
//     String s = scanner.next();
//     subsequences(s,"");
//     System.out.println(result);
//   }
//   static ArrayList<String> result=new ArrayList<String>();
//   static void subsequences(String s,String ans){
//   if(s.length()==0){
//     result.add(ans);
//      return;
//   }
//   subsequences(s.substring(1),ans+s.charAt(0));
//   subsequences(s.substring(1),ans);
//   }
//}
// letter combinations in the phone using the recursion
//import java.util.*;
//public class codes100 {
//   public static void main(String[] args) {
//     Scanner sc=new Scanner(System.in);
//     String a=sc.next();
//     possiblewords(a, "");
```

```
//    }
//
//    static String[] keypad = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
//    static void possiblewords(String a, String ans) {
//        if (a.length() == 0) {
//            System.out.println(ans);
//            return;
//        }
//        String key = keypad[a.charAt(0) - 48]; //mainly used to convert the integer string into
decimal
//        for (int i = 0; i < key.length(); i++) {
//            possiblewords(a.substring(1), ans + key.charAt(i));
//        }
//    }
//}
/*span of the array
span=maximum element-minimum element
 */
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        int n = sc.nextInt();
//        int[] a = new int[n];
//        for (int i = 0; i < a.length; i++) {
//            a[i] = sc.nextInt();
//        }
//        int ans1 = a[0];
//        for (int i = 0; i < n; i++) {
//            if (ans1 < a[i]) {
//                ans1 = a[i];
//            }
//        }
//        int ans = a[0];
//        for (int i = 0; i < n; i++) {
//            if (ans > a[i]) {
//                ans = a[i];
//            }
//        }
//        int span = ans1 - ans;
//        System.out.println(span);
//    }
//}
/*program for rotated sorted array-rotated sorted array means if we have array
```

```
like{1,2,3,4,5,6,7}
if we hae to rotate from index 3 so it looks like {4,5,6,7,1,2,3}*/
//public class codes100 {
//    public static void main(String[] args) {
//        int[] a = {4, 5, 6, 7, 1, 2, 3};
//        int target = 1;
//        System.out.println(rotatedsorted(a, target));
//    }
//
//    static int rotatedsorted(int[] a, int target) {
//        int start = 0;
//        int end = a.length - 1;
//        while (start <= end) {
//            int mid = start + (end - start) / 2;
//            if (target == a[mid]) {
//                return mid;
//            }
//            if (a[start] <= a[mid]) {
//                if (target < a[mid] && target >= a[start]) {
//                    end = mid - 1;
//                } else {
//                    start = mid + 1;
//                }
//            }
//            else {
//                if (target > a[mid] && target <= a[end]) {
//                    start = mid + 1;
//                } else {
//                    end = mid - 1;
//                }
//            }
//        }
//        return -1;
//    }
//}
/*median of two sorted arrays with the complexity of 0(m+n);
median for odd elements=no of elements/2;
median for even elements=sum of middle terms/2
 */
//import java.util.*;
//public class codes100{
//    public static void main(String[] args) {
//        int[] arr1={1,3,8,17};
//        int[] arr2={5,6,7,19,21,23};
```

```
//      System.out.println(mergesorted(arr1,arr2));
//   }
//   static float mergesorted(int[] arr1,int[] arr2) {
//      int i = 0, j = 0, k = 0;
//      int[] mix = new int[arr1.length + arr2.length];
//      while (i < arr1.length && j < arr2.length) {
//         if (arr1[i] < arr2[j]) {
//            mix[k] = arr1[i];
//            i++;
//            k++;
//         } else {
//            mix[k] = arr2[j];
//            j++;
//            k++;
//         }
//      }
//      while (i < arr1.length) {
//         mix[k] = arr1[i];
//         i++;
//         k++;
//      }
//      while (j < arr2.length) {
//         mix[k] = arr2[j];
//         j++;
//         k++;
//      }
//      if (mix.length % 2 == 0) {
//         int mid = mix.length / 2;
//         return (float) (mix[mid] + mix[mid - 1]) / 2;
//      } else {
//         int mid = mix.length / 2;
//         return mix[mid];
//      }
//   }
//}
//rotate the array using java-0(n*k)(it's not an efficient approach)
//import java.util.*;
//public class codes100{
//   public static void main(String[] args) {
//      int[] a = {1, 2, 3, 4, 5};
//      rotatektimes(a,3);
//      for (int i = 0; i < a.length; i++) {
//         System.out.print(a[i] + " ");
//      }
```

```
//    }
//    static void rotatearray(int[] a) {
//        int ans = a[0];
//        for (int i = 1; i < a.length; i++) {
//            a[i - 1] = a[i];
//        }
//        a[a.length - 1] = ans;
//    }
//    static void rotatektimes(int[] a,int k){
//        k=k%a.length;
//        if(k<0){
//            k=k+a.length;
//        }
//        for(int i=1;i<=k;i++){
//            rotatearray(a);
//        }
//    }
//}
//rotate an array with efficient approach-0(n)
//import java.util.*;
//public class codes100{
//    public static void main(String[] args) {
//        int[] a = {1, 2, 3, 4, 5};
//        rotatektimes(a,9);
//        for (int i = 0; i < a.length; i++) {
//            System.out.print(a[i] + " ");
//        }
//    }
//    static void reverseanarray(int[] a,int start,int end){
//         while(start<end){
//             int temp=a[start];
//             a[start]=a[end];
//             a[end]=temp;
//             start++;
//             end--;
//        }
//    }
//    static void rotatektimes(int[] a,int k) {
//        k = k % a.length;
//        if (k < 0) {
//            k = k + a.length;
//        }
//        reverseanarray(a, 0, k - 1);
//        reverseanarray(a, k, a.length - 1);
```

```
//        reverseanarray(a, 0, a.length - 1);
//    }
//}
//move all zeros to the end of the array using two pointer technique
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//        int[] a = {2, 1, 0, 0, 4, 0, 5};
//        zeros(a);
//        for(int i=0;i<a.length;i++){
//            System.out.print(a[i]+" ");
//        }
//    }
//    static void zeros(int[] a) {
//        if (a.length == 0 || a.length == 1) {
//            return;
//        } else {
//            int nz = 0, z = 0;
//            while (nz < a.length) {
//                if (a[nz] != 0) {
//                    int temp = a[nz]; //nz is the non zero and z is the zero
//                    a[nz] = a[z];
//                    a[z] = temp;
//                    nz++;
//                    z++;
//                } else {
//                    nz++;
//                }
//            }
//        }
//    }
//}
//finding the maximum subarray in the array
//subarray-a small part of the array
//import java.util.*;
//public class codes100 {
//    public static void main(String[] args) {
//        int[] a = {1, 2, 3, 4, 5};
//        subarray(a);
//    }
//    static void subarray(int[] a) {
//        for (int i = 0; i < a.length; i++) {
//            for (int j = i; j < a.length; j++) {
//                for (int k = i; k<= j; k++) {
```

```
//                System.out.print(a[k] + " ");
//            }
//            System.out.println();
//        }
//    }
//}
```

```
//import java.util.Scanner;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        int count = 0;
//        for (int i = 0; i < s.length(); i++) {
//            count++;
//        }
//        System.out.println(count);
//    }
//}
```

```
//palindrome strings
//import java.util.Scanner;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        char[] arr = s.toCharArray();
//        System.out.println(palindrome(arr));
//    }
//
//    static boolean palindrome(char[] arr) {
//        int left = 0;
//        int right = arr.length - 1;
//        while (left < right) {
//            if (arr[left] != arr[right]) {
//                return false;
//            } else {
//                left++;
//                right--;
//            }
//        }
//        return true;
//    }
//}
```

```
//anagrams
//import java.util.Arrays;
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String str1 = sc.nextLine(); // eat
//        String str2 = sc.nextLine(); // ate
//        if (str1.length() == str2.length()) {
//            char char1[] = str1.toCharArray();
//            char char2[] = str2.toCharArray();
//            Arrays.sort(char1);
//            Arrays.sort(char2);
//            boolean result = Arrays.equals(char1, char2);
//            if (result) {
//                System.out.println("Anagrams");
//            } else {
//                System.out.println("Not Anagrams");
//            }
//        } else {
//            System.out.println("Not Anagram");
//
//        }
//    }
//}
//java program for the checking whether the given character is uppercase or lowercase
//import java.util.Scanner;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        char ch = sc.next().charAt(0);
//        if (ch >='A' && ch <= 'Z') {
//            System.out.println(1);
//        } else if (ch >= 'a' && ch <= 'z') {
//            System.out.println(0);
//        } else {
//            System.out.println(-1);
//        }
//    }
//}
// calculate the number of vowels and consonants in the given string
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
```

```
//       Scanner sc = new Scanner(System.in);
//       String str = sc.nextLine();
//       int vowelcount = 0, consonantcount = 0,whitespaces=0;
//       for (int i = 0; i < str.length(); i++) {
//          char ch=str.charAt(i);
//          if (ch== 'a'|| ch == 'e'||ch  == 'i' || ch == 'o' || ch == 'u') {
//             vowelcount++;
//          }
//          else if(ch==' '){
//             whitespaces++;
//          }
//          else {
//             consonantcount++;
//          }
//       }
//       System.out.println("number of vowels "+vowelcount);
//       System.out.println("number of consonants "+consonantcount);
//       System.out.println("number of white spaces "+whitespaces);
//    }
//}
//find the ascii value of the character-using type casting we can achieve the following code
//import java.util.Scanner;
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//       Scanner sc = new Scanner(System.in);
//       char ch = sc.next().charAt(0);
//       int result=ch;
//       System.out.println(result); //ascii value of lowercase letter starts from 97 and
uppercase starts from 65
//    }
//}
//remove the vowels from the given string
//import java.util.Scanner;
//import java.util.*;
//public class Stringproblems{
//    public static void main(String[] args) {
//       Scanner sc=new Scanner(System.in);
//       String str=sc.nextLine();
//       String result=str.replaceAll("[aeiouAEIOU]","");
//       System.out.println(result);
//    }
//}
//remove the white spaces from the given string
```

```
//import java.util.Scanner;
//import java.util.*;
//public class Stringproblems{
//    public static void main(String[] args) {
//        Scanner sc=new Scanner(System.in);
//        String str=sc.nextLine();
//        String result=str.replaceAll(" ","");
//        System.out.println(result);
//    }
//}
//program to check whether the given character is alphabet or not
//import java.util.Scanner;
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        char s = sc.next().charAt(0);
//        if (s >= 'A' && s <= 'Z' || s >= 'a' && s <= 'z') {
//            System.out.println(s + " is a alphabet");
//        } else {
//            System.out.println(s + " is not an alphabet");
//        }
//    }
//}
//toggle each character in the string like character is uppercase change it to lowercase
viceversa
//import java.util.Scanner;
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String str=sc.nextLine();
//        String s1="";
//        for(int i=0;i<str.length();i++){
//            if(Character.isUpperCase(str.charAt(i))) {
//                s1 = s1 + Character.toLowerCase(str.charAt(i));
//            }
//            else{
//                s1 = s1 + Character.toUpperCase(str.charAt(i));
//            }
//        }
//        System.out.println(s1);
//    }
//}
```

```
//remove the characters in the string except the alphabets
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        String res=s.replaceAll("[^a-zA-Z]", "");
//        System.out.println(res);
//    }
//}
//Remove the braces from the given algebraic expression
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        String res=s.replaceAll("[{}()]", "");
//        System.out.println(res);
//    }
//}
//count the sum of the digits in the given string
//import java.util.*;
//public class Stringproblems{
//    public static void main(String[] args) {
//        Scanner sc=new Scanner(System.in);
//        String s=sc.nextLine();
//        int sum=0;
//        for(int i=0;i<s.length();i++) {
//            if (s.charAt(i)>='0'&&s.charAt(i)<='9') {
//                sum += (s.charAt(i)-'0');
//            }
//        }
//        System.out.println(sum);
//    }
//}
//captialize the first and last character of the stirng
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        String s1="";
//            s1 = s1+Character.toUpperCase(s.charAt(0))+s.substring(1,s.length()-
1)+Character.toUpperCase(s.charAt(s.length() - 1));
```

```
//          System.out.println(s1);
//    }
//}
//count the frequency of characters in the given string
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        char[] arr = s.toCharArray();
//        int[] freq = new int[256];
//        for (int i = 0; i < arr.length; i++) {
//            freq[arr[i]]++;
//        }
//        for (int i = 0; i < 256; i++) {
//            if (freq[i] > 0) {
//                System.out.println((char)i+ " is " + freq[i]);
//            }
//        }
//    }
//}
//find the first non repeated character in the given string
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        int[] count = new int[256];
//        for (int i = 0; i < s.length(); i++) {
//            count[s.charAt(i)]++;
//        }
//        for (int i = 0; i < s.length(); i++) {
//            if (count[s.charAt(i)] == 1) {
//                System.out.println("the first non repeated character " + s.charAt(i));
//                return;
//            }
//        }
//    }
//}
//replace the substring with another substring
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
```

```
//        String s = sc.nextLine();
//        String oldstring = sc.nextLine();
//        String newstring = sc.nextLine();
//        String result = s.replace(oldstring, newstring);
//        System.out.println(result);
//    }
//}
// replace the word in the given string by another word
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        String oldstring = sc.nextLine();
//        s = s.replaceAll(oldstring,"");
//        s=s.trim();
//        System.out.print(s);
//    }
//}
//java program to print the even number of the characters
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        for (int i = 0; i < s.length(); i++) {
//            if (i % 2 == 0) {
//                System.out.print(s.charAt(i));
//            }
//        }
//    }
//}
//java program to print unique characters
//import java.util.*;
//public class Stringproblems{
//    public static void main(String[] args) {
//        Scanner sc=new Scanner(System.in);
//        String s=sc.nextLine();
//        char[] arr=s.toCharArray();
//        int[] count=new int[256];
//        for(int i=0;i<arr.length;i++) {
//            count[arr[i]]++;
//        }
//        for(int i=0;i<256;i++){
```

```
//          if(count[i]==1) {
//              System.out.print((char)i);
//          }
//      }
//  }
//}
//program to print all the permutations of the given string
//import java.util.*;
//public class Stringproblems{
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        String ans ="";
//        permute(s,ans);
//    }
//    static void permute(String s,String ans){
//        if(s.length()==0) {
//            System.out.println(ans);
//        }
//        else{
//            for(int i=0;i<s.length();i++){
//                String rem=s.substring(0,i)+s.substring(i+1);
//                permute(rem,ans+s.charAt(i));
//            }
//        }
//    }
//}
//pangram strings in java
//import java.util.*;
//public class Stringproblems {
//    public static void main(String[] args) {
//        Scanner sc = new Scanner(System.in);
//        String s = sc.nextLine();
//        System.out.println(pangrams(s));
//    }
//
//    static boolean pangrams(String s) {
//        s = s.toLowerCase();
//        Set<Character> val = new HashSet<>();
//        for (char c : s.toCharArray()) {
//            if (c >= 'a' && c <= 'z') {
//                val.add(c);
//            }
//        }
```

```
//       return val.size() == 26;
//    }
//}
/*convert the string into integer-parseint method only
accepts the string with integer values if we use the character string then it will
throw the number format exception in order ti handle that exception we have to use the try
and catch block keywords that are used in the ecxception handlikng
 */
public class Stringproblems {
   public static void main(String[] args) {
      String s = "1798";
      int result = Integer.parseInt(s);
      System.out.println(s);
   }
}
//
```