class Solution {

    static int[] twoSum(int[] nums, int target) {

     for(int i=0;i <nums.length-1 ; i++)

         for(int j=i+1; j< nums.length ;j++){

             if(nums[i] + nums[j] == target)

              return new int [] {i , j};

         }

          return new int [] {-1,-1};

     }

     public static void main (String [] args){

         int nums[]= {2,7,11,15};

         int target = 9;

          for(int  x : twoSum(nums , target))

         System.out.print(x + " ");

     }

}

class Solution{

          public static void main(String[] args){

             System.out.println(  isPalindrome(121));

          }

    static boolean isPalindrome(int x) {

      int temp = x;

      int r = 0;

      while(x>0){

          int digit = x%10;

          r = (r \*10) +digit;

          x=x/10;

      }

      if (r == temp){

          return true;

      }

      return false ;

    }

}

class Solution {

    public String longestCommonPrefix(String[] v) {

        Arrays.sort(v);

       String s1 = v[0];

       String s2= v[v.length-1];

       int idx = 0;

       while(idx < s1.length() && idx < s2.length()){

           if(s1.charAt(idx) == s2.charAt(idx)){

               idx++ ;

           }

          else{

          break;

          }

       }

     return s1.substring(0 , idx) ;

    }

}

class Solution {

    public boolean isValid(String s) {

    Stack<Character> stack = new Stack<Character>();

    for (char c : s.toCharArray()) {

        if (c == '(')

            stack.push(')');

        else if (c == '{')

            stack.push('}');

        else if (c == '[')

            stack.push(']');

        else if (stack.isEmpty() || stack.pop() != c)

            return false;

    }

    return stack.isEmpty();

}

}

class Solution {

    public int removeElement(int[] nums, int val) {

        int count =0 ,i;

        for( i=0; i< nums.length ; i++){

            if(nums[i] != val)

            nums[count++] = nums[i];

        }

        return count;

    }

}

class Solution

 {

    public int strStr(String haystack, String needle)

    {

        for(int i=0; i+needle.length() <= haystack.length(); i++)

        {

            if(haystack.substring(i,i+needle.length()).equals(needle)){

                return i;

            }

        }

        return -1;

    }

}

class Solution {

    public int removeDuplicates(int[] nums) {

        int count = 0;

        // Loop for all the elements in the array

        for (int i = 0; i < nums.length; i++) {

            if ( i < nums.length - 1 && nums[i] == nums[i + 1]) {

                continue;

            }

            // We will update the array in place

            nums[count] = nums[i];

            count++;

        }

        return count;

    }

}

class Solution {

    public void merge(int[] nums1, int m, int[] nums2, int n) {

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

            for (int j= 0; j < n ; j++){

               nums1[i] = nums2[j] ;

               i++;

            }

            Arrays.sort ( nums1 );

        }

    }

}

class Solution {

    public boolean isValid(String s) {

    Stack<Character> stack = new Stack<Character>();

    for (char c : s.toCharArray()) {

        if (c == '(')

            stack.push(')');

        else if (c == '{')

            stack.push('}');

        else if (c == '[')

            stack.push(']');

        else if (stack.isEmpty() || stack.pop() != c)

            return false;

    }

    return stack.isEmpty();

}

}