public class yelp {

  public String getCompressedString(String s)

  {

      if(s.length() == 0) return s;

      StringBuilder result = new StringBuilder();

      for(int i = 0; i < s.length();)

      {

          char current = s.charAt(i);

          int count = 0;

          while(i< s.length() && s.charAt(i) == current)

          {

              i++;

              count++;

          }

          result.append(count);

          result.append(current+"");

      }

      return result.toString();

  }

  public boolean checkPalindrome(String s)

  {

      if(s.length() == 0) return true;

      int start = 0, end = s.length() - 1;

      while (start<end)

      {

          if(s.charAt(start) != s.charAt(end))

          {

              return false;

          }

          start++;

          end--;

      }

      return true;

  }

  public String[] topColor(String[][] colors)

  {

      ArrayList<String> res = new ArrayList<>();

      int m = colors.length;

      int n = colors[0].length;

      HashMap<String, Integer> map = new HashMap<>();

      int maxCount = 0;

      for(int i = 0; i < m; i++)

      {

          for(int j = 0; j < n ; j++)

          {

              int currentCount = 0;

              if(map.containsKey(colors[i][j]))

              {

                  currentCount =  map.get(colors[i][j]);

                  map.put(colors[i][j], ++currentCount);

              }else

              {

                  map.put(colors[i][j], 1);

                  currentCount = 1;

              }

              maxCount = Math.*max*(maxCount, currentCount);

          }

      }

      for(String key : map.keySet())

      {

          if(map.get(key) == maxCount)

          {

              res.add(key);

          }

      }

      Collections.*sort*(res);

      return res.toArray(new String[res.size()]);

  }

  public ArrayList<String> topColorList(ArrayList<List<String>> colors)

  {

      ArrayList<String> res = new ArrayList<>();

      if(colors.size() == 0) return res;

      HashMap<String, Integer> map = new HashMap<>();

      int maxCount = 0;

      for(List<String> list : colors)

      {

          for(String color : list)

          {

              int currentCount = 0;

              if(map.containsKey(color))

              {

                  currentCount =  map.get(color);

                  map.put(color, ++currentCount);

              }else

              {

                  map.put(color, 1);

                  currentCount = 1;

              }

              maxCount = Math.*max*(maxCount, currentCount);

          }

      }

      for(String key : map.keySet())

      {

          if(map.get(key) == maxCount)

          {

              res.add(key);

          }

      }

      Collections.*sort*(res);

      return res;

  }

  public String findRestaurants(List<String> restaurants\_1, List<String> restaurants\_2)

  {

      HashMap<String, Integer> map = new HashMap<>();

      for (int i = 0; i < restaurants\_1.size(); i++)

      {

          map.put(restaurants\_1.get(i), i);

      }

      int minRank = Integer.*MAX\_VALUE*;

      String res = null;

      for(int i = 0; i < restaurants\_2.size(); i++)

      {

          if(map.containsKey(restaurants\_2.get(i)))

          {

              int rank = map.get(restaurants\_2.get(i)) + i;

              if(rank < minRank)

              {

                  minRank = rank;

                  res = restaurants\_2.get(i);

              }

          }

      }

      return res == null ? "Yelpwich" : res;

  }

  public String rmE (String s) {

      StringBuilder sb = new StringBuilder();

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

          if (s.charAt(i) == 'e') {

              sb.append(s.charAt(i));

              while(i < s.length() && s.charAt(i) == 'e')

              {

                  i++;

              }

          }

          else {

              sb.append(s.charAt(i));

              i++;

          }

      }

      return sb.toString();

  }

  public List<Integer> findId(List<BusinessInfo> list) {

      List<Integer> ids = new ArrayList<>();

      for (BusinessInfo i : list) {

          String str = i.catagory;

          List<String> categories = Arrays.*asList*(str.replaceAll("\\s+", "").split(","));

          if (categories.contains("japanese") && categories.contains("resturant")) {

              ids.add(i.id);

          }

      }

      return ids;

  }

  public Double medianRating(List<BusinessInfo> list)

  {

      Comparator<BusinessInfo> ratingComp = new Comparator<BusinessInfo>() {

          @Override

          public int compare(BusinessInfo o1, BusinessInfo o2) {

              return o1.rating - o2.rating < 0 ? 1 : -1;

          }

      };

      Collections.*sort*(list,ratingComp);

      if(list.size() % 2 != 0)

      {

          return list.get(list.size() / 2).rating;

      }else {

          Double r1 = list.get(list.size() / 2).rating;

          Double r2 = list.get(list.size() / 2 - 1).rating;

          return (r1+r2)/2;

      }

  }

}

242. Valid Anagram

/\*\*

\* @param {string} s

\* @param {string} t

\* @return {boolean}

\*/

var isAnagram = function(s, t) {

   if(s==null&&t==null) return true;

   if(s==null||t==null) return false;

   if(s.length!=t.length) return false;

   var hash = new Map();

   for(let i=0; i<s.length; i++) {

       if(!hash.has(s[i])) {

           hash.set(s[i],1);

       } else {

           hash.set(s[i], hash.get(s[i])+1);

       }

   }

   for(let i=0; i<t.length; i++) {

       if(hash.has(t[i])) {

           hash.set(t[i], hash.get(t[i])-1);

           if(hash.get(t[i])<=0) {

               hash.delete(t[i]);

           }

       }else {

           return false;

       }

   }

   return hash.size==0 ? true: false;

};