```
# ============Q1============
class Solution:
    def isIsomorphic(self, s: str, t: str) -> bool:
        return len(set(s))==len(set(zip(s,t)))==len(set(t))

    #============Q2================
    class Solution {
public:
    bool isStrobogrammatic(string num) {
        int len = num.size();
        for (int i = 0; i < len; ++ i) {
            switch (num[i] - 48) {
                case 2:
                case 3:
                case 4:
                case 5:
                case 7: return false;
                case 6: if ('9' != num[len - 1 - i]) return false; break;
                case 9: if ('6' != num[len - 1 - i]) return false; break;
                case 1:
                case 8:
                case 0: if (num[i] != num[len - 1 - i]) return false;
break;
            }
        }
        return true;
    }
}

    # ============Q3============#

    class Solution:
    def addStrings(self, num1: str, num2: str) -> str:

        def str2int(num):
            numDict = {'0' : 0, '1' : 1, '2' : 2, '3' : 3, '4' : 4, '5' :
5,
                  '6' : 6, '7' : 7, '8' : 8, '9' : 9}
            output = 0
            for d in num:
                output = output * 10 + numDict[d]
            return output

        return str(str2int(num1) + str2int(num2))

# ===============Q4==============#

class Solution:
def reverseWords(self, s: str) -> str:
    return ' '.join(word[::-1] for word in s.split())

# ==============Q5==========#
def reverseStr(self, s: str, k: int) -> str:
    s=list(s)
    for i in range(0,len(s),2*k):
        s[i:i+k]=(s[i:i+k])[::-1]
    return ''.join(s)


# ==========Q6=========#

class Solution(object):
def rotateString(self, s, goal):
    """
    :type s: str
    :type goal: str
    :rtype: bool
    """
    if(len(s)!=len(goal)):
        return False
    for i in range(len(s)):
        if(s==goal):
            return True
        else:
            s = s[1:]+s[0]
    return False

# ===========Q7===========#
class Solution:
```

```python
def backspaceCompare(self, s: str, t: str) -> bool:
    new_s = new_t = ""
    for letter in s:
      if letter != '#':
        new_s += letter
      else:
        new_s = self.remove_previous(new_s)
    for letter in t:
      if letter != '#':
        new_t += letter
      else:
        new_t = self.remove_previous(new_t)

    if new_s == new_t:
      return True
    return False

def remove_previous(self,s):
  if s == "":
    return s
  return s[:-1]

# ===========Q8===========#

def checkStraightLine(self, coordinates: List[List[int]]) -> bool:
(x1, y1), (x2, y2) = coordinates[0], coordinates[1]

for x3, y3 in coordinates[2:]:
    if (y2 - y1) * (x3 - x1) != (y3 - y1) * (x2 - x1):
        return False

return True
```