Q1:

```
public int scoreGuess(String guess)

{

    int count = 0;

    for (int i = 0; i <= secret.length() - guess.length(); i++)

     {

        if (secret.substring(i, i + guess.length()).equals(guess))

        {

            count++;

        }

    }

    return count * guess.length() * guess.length();

}

public String findBetterGuess(String guess1, String guess2)

 {

    if (scoreGuess(guess1) > scoreGuess(guess2))

    {

        return guess1;

    }

    if (scoreGuess(guess2) > scoreGuess(guess1))

    {
        return guess2;
    }
    if (guess1.compareTo(guess2) > 0)
    {
        return guess1;
    }
    return guess2;
}
```

Q2:

```
public class CombinedTable
{
  private SingleTable table1;
  private SingleTable table2;
  public CombinedTable(SingleTable table1, SingleTable table2)
   {
    this.table1 = table1;
    this.table2 = table2;
  }

  public boolean canSeat(int n) {
    if (table1.getNumSeats() + table2.getNumSeats() - 2 >= n)
     {
      return true;
    } else {
      return false;
    }
  }
  public double getDesirability()
  {
    if (table1.getHeight() == table2.getHeight())
    {
      return (table1.getViewQuality() + table2.getViewQuality()) / 2;
    }
     else
     {
      return (table1.getViewQuality() + table2.getViewQuality()) / 2 -
10;
    }
  }
}
```

Q3:

```
public void addMembers(String[] names, int gradYear)
 {
    for (String n : names)
     {
        MemberInfo newM = new MemberInfo(n, gradYear, true);
        memberList.add(newM);
    }
}
public ArrayList<MemberInfo> removeMembers(int year)
{
    ArrayList<MemberInfo> removed = new
ArrayList<MemberInfo>();
    for (int i = memberList.size() - 1; i >= 0; i--)
     {
        if (memberList.get(i).getGradYear() <= year)
        {
            if (memberList.get(i).inGoodStanding())
             {
                removed.add(memberList.get(i));
            }
            memberList.remove(i);
        }
    }
    return removed;
}
```

Q4:

```
public static boolean isNonZeroRow(int[][] array2D, int r)
 {
  // Iterate over the columns in the row.
  for (int col = 0; col < array2D[0].length; col++)
  {
    if (array2D[r][col] == 0) {
      return false;
    }
  }
  // If no zeros are found, return true
  return true;
}
public static int[][] resize(int[][] array2D)
 {
  int numRows = array2D.length;
  int numCols = array2D[0].length;
  int numNonZeroRows = 0;
  for (int r = 0; r < numRows; r++)
  {
    if (isNonZeroRow(array2D, r))
    {
      numNonZeroRows++;
    }
  }
 int[][] result = new int[numNonZeroRows][numCols];
  int newRowIndex = 0;
  for (int r = 0; r < numRows; r++)
  {
    if (isNonZeroRow(array2D, r))
    {
      for (int c = 0; c < numCols; c++)
      {
        result[newRowIndex][c] = array2D[r][c];
      }
      newRowIndex++;
    }
  }
  return result;
}
```

_ 1:

```
______ ___ __________(______ _____)
{
   ___ _____ = 0;
   ___ (___ _ = 0;_ <= ______.______() - _____._____
_ ();_ ++)
    {
      __ (_____.________(_ ,_ + _____.______()).__
____(_____))
      {
         _____++;
      }
   }
   ______ _____ *_____.______() *_____.______();
}
______ ______ _____________(______ _____1, _
_____ _____2)
 {
   __ (_________(_____1) > _________(_____2))
   {
      _____ _____1;
    }
    __ (__________(_____ 2) > __________(_____ 1))
    {
      ______ _____ 2;
    }
    __ (_____ 1._________(_____ 2) > 0)
    {
      ______ _____ 1;
    }
    ______ _____ 2;
}


_ 2:

______ _____ _____________
{
  _______ ___________ _____ 1;
  _______ ___________ _____ 2;
  ______ _____________(___________ _____ 1, ___
________ _____ 2)
   {
    ____._____ 1 = _____ 1;
  ____._____2 = _____2;
 }


 _____ ______ _______(___ _){
  __ (_____1.___________() + _____2.___________()
- 2 >= _)
    {
     _____ ____;
   }____ {
     _____ _____;
   }
 }
 _____ _____ ______________()
 {
  __ (_____1.__________() == _____2.__________())
   {
     ______ (_____1._____________() + _____2.____
___________())/ 2;
   }
   ____
    {
```

```
   ______ (____ 1.______________ () + _____ 2.____
___________ ()) / 2 - 10;
    }
  }
}


_ 3:

_____ ____ __________ (_____ []_____ , ___ ____
____ )
 {
    ___ (______ _ :_____ )
     {
        __________ ____  = ___ __________ (_ , _______
_ , ____);
        __________ .___ (____);
    }
}
_____ _________ <_________ > ____________
(___ ____ )
{
    _ _ _ _ _ _ _ _ <_ _ _ _ _ _ _ _ _ _ > _ _ _ _ _ _ _  = _ _ _  _ _ _ _ _ _ _
_ _ <_ _ _ _ _ _ _ _ _ _ >();
    _ _ _  (_ _ _ _  = _ _ _ _ _ _ _ _ _ _ ._ _ _ _ () - 1;_  >= 0;_ --)
     {
        _ _  (_ _ _ _ _ _ _ _ _ ._ _ _ (_ )._ _ _ _ _ _ _ _ _ _ _ () <= _ _ _ _ )
        {
            _ _  (_ _ _ _ _ _ _ _ _ ._ _ _ (_ )._ _ _ _ _ _ _ _ _ _ _ _ _ _ ())
            {
                _ _ _ _ _ _ _ ._ _ _ (_ _ _ _ _ _ _ _ _ ._ _ _ (_ ));
            }
            _ _ _ _ _ _ _ _ _ _ ._ _ _ _ _ _ (_ );
        }
    }
    _ _ _ _ _ _  _ _ _ _ _ _ _;
}

_ 4:

_ _ _ _ _ _  _ _ _ _ _ _  _ _ _ _ _ _ _  _ _ _ _ _ _ _ _ _ _ _ _ (_ _ _ [][] _ _ _ _
_ 2_ , _ _ _  _ )
 {
  // _ _ _ _ _ _ _  _ _ _ _  _ _ _  _ _ _ _ _ _ _  _ _  _ _ _  _ _ _ .
  _ _ _  (_ _ _ _ _ _  = 0;_ _ _  < _ _ _ _ _ 2_ [0]._ _ _ _ _ _ ;_ _ _ ++)
  {
    __ (_____2_[_][___] == 0){
      ______ _____;
    }
  }
  //__ __ _____ ___ _____,______ ____
  ______ ____;
}
______ ______ ___[][] ______(___[][] _____2_)
 {
  ___ _______ = _____2_._______;
  ___ _______ = _____2_[0]._______;
  ___ _____________ = 0;
  ___ (___ _ = 0;_ < _______;_++)
  {
    __ (___________(_____2_,_))
     {
      _____________++;
    }
  }
 ___[][] ______ = ___ ___[_____________][______
_];
```

```
 ___ ___________ = 0;
 ___ (___ _ = 0;_ < _______;_ ++)
 {
  __ (___________(_____ 2_ , _))
  {
   ___ (___ _ = 0;_ < _______;_ ++)
   {
    ______[___________][_] = _____ 2_[_][_];
   }
   ___________++;
  }
 }
 ______ ______;
}
```