2018-AP:CSA FRQ

```
Public boolean simulate(){
        int temp=0;
        for(int i=0;i<goalDistance;i++){
            temp+=hop.Distance();
            if(temp<0) return false;
            if(temp >= goalDistance)return true;
        }
    return false;
}


Public double runSimulations(int num){
      int right=0;
      for(int i=0; i<num ;i++){
          if(simulate()){
              right++;
          }
      }
      return ((double)right)/num;
}

Public  WordPairList(String[] words) {
      allPairs = new ArrayList<WordPair>();
      for(int i=0;i< words.length-1 ;i++){
          for(int j=i+1;j<words.length;j++){
              allPairs.add(new WordPair(words[i],words[j]));
          }
      }
}

 Public int numMatches(){
      int temp=0;
      for(WordPair e: allPairs){
            if(e.getFirst().equals(e.getSecond())) temp++;
      }
      return temp;
}
```

```
Public class CodewordChecker implements StringChecker {
        Private int min,max;
        Private string delim;

        Public CodewordChecker(int min, int max, string delim) {
              this.min = min;
              this.max = max;
              this.delim=delim;
        }
        Public CodeWordChecker(String delim){
              this(6,20, delim);
        }
        Public boolean isValid (String str){
              if(str.length() < min || str.length >max) {
                    return false;
              }
              for(int i =0; i < str.length(); i++) {   // couldve used indexof
                    if(str.charAt(i).equals(delim))
                          return false;
              }
              return true;
        }
}

Public static int[] getColumn(int[][] arr2D, int c) {
       int[] arr = new int[arr2D.length]; // arr2D[x] -> col length
       for(int i=0; i < arr2D.length; i++){
           arr[i] = arr2D[i][c];
       }
       return arr;
}

Public static boolean isLatin(int[][] squares){
       if(containsDuplicates(squares[0]))
           return false;
       for(int i = 1; i < squares.length; i++){
            if(!hasAllValues(squares[0], squares[i]))
                 return false;
       }
       for(int i=0; i < squares[0].length; i++) {
           if(!hasAllValues(square[0], getcolumn(squares, i)))
                return false;
       }
       return true;
}
```

2019 AP: CSA FRQ

```
Public static int numberOfLeapYears (int year1, int year2) {
        int t=0;
        for(int i = year1; i <= year2; i++) {
            if (isLeapYear(i))
              t++;
        }
        return t;
}

Public class StepTracker {
      Private int min, days, steps, aDays;
      Public StepTracker(int min){
           this. min = min;
           days=0;
           steps =0;
           aDays=0;
      }
      Public void addDailySteps(int s) {
            if(s>=min){
              aDays++;
            }
           steps +=s;
           days++;
      }
      Public int activeDays() {
            return aDays;
      }
      Public double averageSteps(){
           //return ((double) steps / days; check division by 0
           if(days == 0)
              return 0;
           else
              return ((double) steps) / days;
      }
}

Public ArrayList<String> getDelimitersList(String[] tokens) {
      ArrayList<String> arg = new ArrayList<String>();
      for(String tok : tokens) {
           if(tok.equals(openDel) || tok.equals(closeDel)) {
                 arg.add(tok);
           }
      }
      return arg;
}
```

```
Public boolean isBalanced(ArrayList<String> delimiters){
        int k=0, c=0;
        for(String token: delimiters){
           if(token.equals(openDel)){ //LMAO stop overcomplicating, didn't read conditions
                k++;
           else
                c++;
           if(c>k) return false;
        }
        return c==k;
   }

   Public LightBoard(int numRows, int numCols){
        lights = new boolean[numRows][numCols];
        for(int i=0; i<numRows; i++){
           for(int j=0; j<numCols; j++){
                lights[i][j] = Math.random()<0.4;
           }
        }
   }

   public boolean evaluateLight(int row, int col){
         boolean rLight = lights[row][col];
         if(rLight){
           int on=0;
           for(int i=0; i<lights.length; i++){
                if(lights[ i ][col]){
                     on++;
                }
           }
           if(on%2==0){
             return false;
           }
         }else{
           int on=0;
           for(int i=0; i<lights.length; i++){
                if(lights[ i ][col])
                     on++;
           }
           if(on%3==0){
             return true
           }
         }
         return rLight;
      }
```

2017 - AP: CSA FRQ

```
Public Digits(int num) {
     digitList = new ArrayList<Integer>();
     if(num == 0)
       digitList.add(0);
     else {
          while(num > 0) {
             int curr=num % 10;
             digitList.add(curr);
             num /= 10;
          }
     }
}

Public boolean isStrictlyIncreasing() {
      int last = digitList.get(0);
      if(digitList.size() == 1)
        return true;
      for(int i=1; i< digitList.size(); i++) {
          if(last >= digitList.get(i) ) {
             return false;
          }
          last = digitList.get(i);
      }
      return true;
  }

Public class MultPractice implements StudyPractice {
       Private int constVal, incVal;
       Public MultPractice(int boundConst, int boundInc) {
             this.constVal = boundConst;
             this.incVal = boundInc;
       }
       Public void nextProblem() {
             incVal++;
       }
       Public String getProblem() {
            Return constVal + " TIMES " + incVal;
       }
   }

   Public void replaceNthOccurrence(String str, int n, String repl) {
             int dex = findNthOccurrence(str, n, repl);
             String temp = "";
             if(dex != -1) {
              currentPhrase=currentPhrase.substring(0, dex) + repl+ currentPhrase.subString(dex + str.length());
             }
   }
   Public int findLastOccurrence(String str) {
         int k=0;
         while(findNthOccurrence(str, k+1) != -1) {
              k++;
         }
         return k;
   }
```

```
Public static Position findPosition(int num, int[][] intArr){
    for(int i=0; i<intArr.length; i++){
        for(int j=0; j<intArr[0].length; j++){
            if(intArr[i][j]==num){
                return new Position(i,j);
            }
        }
    }
    return null;
}

Public static Position[][] getSuccessorArray(int[][] intArr){
    Position[][] arr = new Position[intArr.length][intArr[0].length];
    for(int i=0; i<intArr.length; i++){
        for(int j=0; j<intArr[0].length; j++){
            arr[i][j] = findPosition(intArr[i][j]+1, intArr);
        }
    }
    return arr;
}
```

2018- APCSA MCQ

1. E
2. E
3. B
4. C
5. A
6. C
7. D
8. A
9. B
10. E
11. E
12. A
13. D
14. B
15.

2021 - AP:CSA FRQ

#3a

```
public void addMembers(String[] names, int gradYear){
    for(String str: names){
        memberList.add(new MemberInfo(str, gradYear, true));
    }
}
```

#3b

```
Public ArrayList<MemberInfo> removeMembers(int year) {
        ArrayList<MemberInfo> temp = new ArrayList<MemberInfo>();
        for(int i=0; i<memberList.size(); i++) {
            MemberInfo x = memberList.get(i);
            if(x.getGradYear() <= year){
                if(x.inGoodStanding()){
                    temp.add(x);
                }
                memberList.remove(i);
            }                              ← i--
        }
        return temp;
    }
```

#4a

```
Public static boolean isNonZeroRow(int[][] array2D, int r) {
      for(int i=0; i<array2D[r].length; i++){
          if(array2D[r][i] == 0){
              return false;
          }
      }
      return true;
}
```

#4b

```
Public static int[][] resize(int[][] array2D) {
      int[][] temp = new int[numZeroRows(array2D)][array2D[0].length];
                                                  <newRowIndex : int = 0
      for(int i=0; i<array2D.length; i++){
          if(isNonZeroRow(array2D,i)){
              for(int j=0; j<array2D[i].length; j++){
                  temp[i][j] = array2D[i][j];
              }
          }                    <newRowIndex++
      }                        newRowIndex
      return temp;
}
```

2017 FRQ

#1a

```
Public Digits(int num) {
      digitList = new ArrayList<Integer>;
      while(num != 0) {
          digitList.add(num % 10);
          num /= 10;
      }
}
```

#1b
```
public boolean isStrictlyIncreasing() {
    int last;
    for(int i = 0; i < digitList.size(); i++) {
        if(i==0) {
            last = digitList.get(i);
        } else {
            if(digitList.get(i) <= last) {
                return false;
            }
        }
    }
    return true;
}
```

#2
```
public class MultPractice implements StudyPractice {
    private int a, b;

    Public MultPractice(int constVar, int incVar) {
        this.a = constVar;
        this.b = incVar;
    }

    Public String getProblem() {
        return a + " TIMES " + b;
    }
    public void nextProblem() {
        b++;
    }
}
```

#3a
```
public void replaceNthOccurrence(String str, int n, String repl)
    int start = findNthOccurrence(str, n);
    String prev = currentPhrase(0, start-1);
    String conc = current
```

Scratchpad

e(8): 144

e(8-2) × e(8-1) 1728

e(6-2) × e(6-1) e(7-2) × e(7-1)

12 × 12 12 e(6-2) × e(6-1)

144 12 × 12

144

144 × 12 = 1728

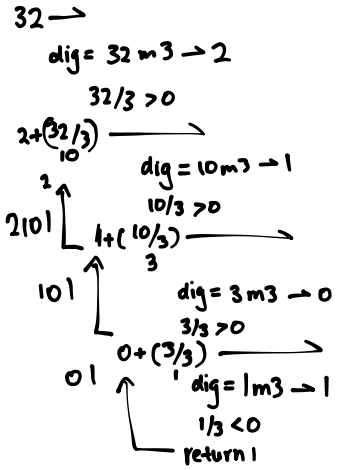
144
× 12
288
1440
1728

2 | 3
2 | 3
1728
× 144
176912
69120
172800
248832

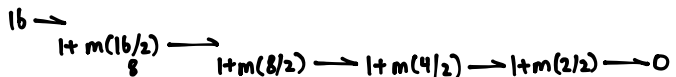
r(7):
r(7-2):
r(5-2):
r(3-2):
r(1-2):
-1
StackOverflow

(1) (2) (3) (4) (5)
today + oday + day + ay + y

+ o d a y
0 1 2 3 4 (E)



123456



2013 - APCSA - FRQ

```
Public DownloadInfo getDownloadInfo(String title){
        DownloadInfo temp= null;
        for(DownloadInfo e: download List){
            if(e.getTitle().equals(title)){
                temp = e;
                break;
            }
        }
        return temp;
}

Public void updateDownloads(List<String> titles) {
        for(int i =0; i< titles. size(); i++){
            boolean found =false;
            for(int j=0; j < download List. size(); j++){
                if(!found) {
                    if(download List.get(j).getTitle().equals(titles.get(i)) {
                        download List.get(j). incrementTimesDownloaded();
                        found =true;
                    }
                }
            }
            if(!found) {
                download List.add(new DownloadInfo( titles.get(i)));
            }
        }
}

    Public TokenPass(int playerCount) {
        board = new int[playerCount];
        for(int i =0; i< board.length; i++){
            board[i] = (int) (Math.random() *(10)) +1;
        }
        currentPlayer = (int) (Math.random() *(playerCount));
    }
```