Does God play Tetris? - Team reference document

Program submission checklist:

1. Works on sample inputs given.
2. Works on other sensible inputs.
3. Works on pathalogical inputs/corner cases.
4. Works in time on the largest possible inputs.
5. Compiles! (with warnings on! -Xlint)
6. No debug outputs!

Code

Big sample

```java
import java.io.*;
import java.util.*;
import java.math.*;
public class samplecode
{
  public static void debug(String s) {
    System.out.printf(">>>%s>>>\n", s); //Comment this out to kill n birds with two /
  }
  public static void main(String[] args) throws Exception {
    // Read in input:
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    String s1 = br.readLine();
    int a = Integer.parseInt(s1.split(" ")[0]);
    String[] arr = s1.split(" ");

    // Does God play Tetris? used java.util.Collections, it's super effective!

    // A comparator can be defined by
    class MyClassCmp implements Comparator<MyClass> {
      // Should return a negative integer, zero, or a positive integer as the first
          argument is less than, equal to, or greater than the second respectively
      public int compare(MyClass a, MyClass b) {
        return a.a - b.a;
      }

      // As far as I can tell this may not be neccessary, but probably best to do anyway
      public boolean equals(MyClass a, MyClass b) {
        return a.a == b.a;
      }
    }
    // To change an array to a list we can do
    List<String> arrayaslist = Arrays.asList(arr);
    // Or make a general list
    List<MyClass> list = new LinkedList<MyClass>();
    List<MyClass> list2 = new Vector<MyClass>();
    // If we have a comparator already we can do
    Collections.sort(arrayaslist);
    // or maybe
    Collections.sort(list, new MyClassCmp());
    // If we have a sorted list we can do
    MyClass target = new MyClass(3);
    Collections.binarySearch(list, target, new MyClassCmp());
    SortedSet<MyClass> set = new TreeSet<MyClass>(new MyClassCmp());

    // We can work with arbitrary precision integers as follows:
```

```
    BigInteger numb = new BigInteger("12234237843295458912384712938123912546 51");
    numb = numb.add(BigInteger.valueOf(3));
    debug(numb.toString());

    // In places where code should never be reached we can debug (and submit) with assert(false); there, this way we will get an exception rather than dodge behaviour.
    debug(arr[0]);
  }

  // Custom classes declared within the main like this:
  static class MyClass {
    int a;
    MyClass(int A) {
      a= A;
    }
  }
}
```

Graphs

Max-Flow

Shortest path

Min spanning tree

Number theory

GCD

```
public class gcd {
  static int gcd(int a, int b) {
    int c = 0;
    while(a!=0 && b!=0) {
      c = b;
      b = a%b;
      a = c;
    }
    return a+b;
  }

  static int arrGCD(int[] a) {
    int g = a[0];
    for (int i = 0; i < a.length; i++) {
      g = gcd(a[i],g);
      if (g == 1) break;
    }
    return g;
  }
}
```

$$\operatorname{lcm}(a,b) = ab/\gcd(a,b)$$

Dynamic programming

Discrete knapsack problem

Combinatorics

Derangements, permutations, other bits

Logic

2-SAT (requires strongly connected components??)

String algorithms

Matching

```
public class kmp {
  static int[] preKmp(char[] x, int m, int[] kmpNext) {
    int i, j;

    i = 0;
    j = kmpNext[0] = -1;
    while (i < m) {
      while (j > -1 && x[i] != x[j])
        j = kmpNext[j];
      i++;
      j++;
      if (x[i] == x[j])
        kmpNext[i] = kmpNext[j];
      else
        kmpNext[i] = j;
    }
    return kmpNext;
  }

  static void KMP(char[] x, int m, char[] y, int n) {
    int i, j;
    int[] kmpNext = new int[x.length];

    /* Preprocessing */
    kmpNext = preKmp(x, m, kmpNext);

    /* Searching */
    i = j = 0;
    while (j < n) {
      while (i > -1 && x[i] != y[j])
        i = kmpNext[i];
      i++;
      j++;
      if (i >= m) {
        System.out.println(j - i);
        i = kmpNext[i];
      }
    }
  }
}
```

Geometric algorithms

Simple data structures

```
public class Point implements Comparable<Point> {
  int x; int y;
  public int compareTo(Point p) {return (x-p.x == 0) ? y-p.y : x-p.x;}// left-bottommost
  public float cross(Point p) { return x*p.y - p.x*y; }
}
```

Convex hull (can be used for furthest points)

```
import java.util.*;
public class convexhull
{
  static final double eps = 0.0000000001;
  static int isAnti(Point x0,Point x1,Point x2) {
    double a = (x1.x-x0.x)*(x2.y-x0.y)-(x2.x-x0.x)*(x1.y-x0.y);
    if (a > eps || -a > eps) return a > 0 ? -1 : 1;
    return 0;
  }
  static int isCloser(Point x0,Point x1,Point x2) {
    double d1 = (x0.x - x1.x)*(x0.x - x1.x) + (x0.y - x1.y)*(x0.y - x1.y);
    double d2 = (x0.x - x2.x)*(x0.x - x2.x) + (x0.y - x2.y)*(x0.y - x2.y);
    if (d1-d2 > eps || d2-d1 > eps) return d1 < d2 ? -1 : 1;
    return 0;
  }
  public static List<Point> hull(List<Point> points) {
    Collections.sort(points);
    final Point p0 = points.get(0);
    points.remove(p0);
    Collections.sort(points, new Comparator<Point>() {
        public int compare(Point p1, Point p2) {
        int a = isAnti(p0,p1,p2);
        if (a != 0) return a;
        return isCloser(p0,p1,p2);
        }});
    int m = points.size();
    for (int i = 1; i < m; i++) { // Remove colinears
      if (isAnti(p0,points.get(i-1),points.get(i)) == 0) {
        points.remove(i-1);
        m--;
      }
    }
    LinkedList<Point> hull = new LinkedList<Point>();
    if (m < 2) return hull; // All colinear, no hull
    hull.push(p0);
    hull.push(points.get(0));
    hull.push(points.get(1));
    for (int i = 2; i < m; i++) {
      while (isAnti(hull.get(0),hull.get(1),points.get(i))<=0) {
        hull.pop();
      }
      hull.push(points.get(i));
    }
    return hull;
  }
}
```

Closest pair of points

```
import java.util.*;
public class closestpoints {
  public static Point[] closestPair(Point[] arr){
    Point[] ret = {arr[0],arr[1]};
    Arrays.sort(arr);
    return ret;
  }
}
```