

# 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 pathological 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

```
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 necessary, 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("1223423784329545891238471293812391254651");
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;
    }
}
}

```

## Graph algorithms

Max-Flow

Shortest path

Min spanning tree

## 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];
            }
        }
    }
}

```

}  
}  
}

## 0.1 Geometric algorithms

Convex hull (can be used for furthest points)

Closest 2d pair points