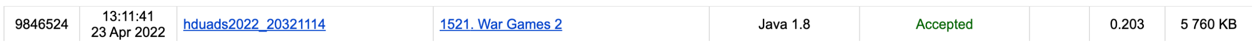
Laboratory work # 5

Student: HU Riqian
Student ID: 20321114
Timus Name: hduads2022_20321114

Mail: jhlxhrq@163.com

Problem # 1521. War Games 2

Screenshot from Timus:



9846524	13:11:41 23 Apr 2022	hduads2022_20321114	1521. War Games 2	Java 1.8	Accepted		0.203	5 760 KB

Explanation of algorithm:

1. I regard this question as Josephus problem.

2. I come up with this recursive idea: continuously + k % n to judge who is the next one to go.

3. Use the Binary Indexed Trees to keep the total index ahead of the people who go away.

Computational complexity of algorithm:

$$O(N\log^2 N)$$

Source code:

```
import java.io.*;

public class WarGame2 {
    public static void main(String[] args) throws IOException {
        new WarGame2().run();
    }

    static StreamTokenizer in;
    static PrintWriter out;
    static int N;
    static int K;
    static int[] CIRCLE;

    int nextInt() throws IOException {
        in.nextToken();
        return (int) in.nval;
    }
```

```java
    void run() throws IOException {
        in = new StreamTokenizer(new BufferedReader(new
InputStreamReader(System.in)));
        out = new PrintWriter(System.out);
        inputData();
        solve();
        out.flush();
    }

    void inputData() throws IOException {
        N = nextInt();
        K = nextInt();
        CIRCLE = new int[N + 1];
    }

    void solve() {
        for (int i = 1; i <= N; i++) {
            addToArray(i, 1);
        }
        int c = 1;
        int index = N;
        int temp;
        for (int i = 1; i <= N; i++) {
            c = (c + K - 1) % index;
            if (c == 0) {
                c = index;
            }
            temp = search(c, N);
            addToArray(temp, -1);
            index--;
        }
    }
    static void addToArray(int i, int x) {
        while (i <= N) {
            CIRCLE[i] += x;
            i += (i & (-i));
        }
    }
    static int indexTotal(int i) {
        int s = 0;
        while (i > 0) {
            s += CIRCLE[i];
            i -= (i & (-i));
        }
        return s;
    }
    static int search(int x, int n) {
        int l = 1;
        int m, t;
        while (l <= n) {
            m = (l + n) / 2;
            t = indexTotal(m);
            if (t < x) {
                l = m + 1;
            } else {
                n = m - 1;
            }
        }
        out.print(l + " ");
        return l;
    }
}
```