import java.util.Random;

public class TeamStats extends Thread {

final private int id;

static int[] array;

static int[][] teamwins;

static int size;

static int numteams;

static int numthreads;

static int set;

int maxwins = 0;

int maxteam = 0;

public TeamStats(int id) {

this.id = id;

}

public static void main(String[] args) throws InterruptedException {

//Parse parameters

size = Integer.parseInt(args[0]);

numteams = Integer.parseInt(args[1]);

numthreads = Integer.parseInt(args[2]);

set = size / numthreads;

//Create necessary variables

array = new int[size];

teamwins = new int[numthreads][numteams];

Random r = new Random();

TeamStats[] threads = new TeamStats[numthreads];

//Initialize array with random numbers between 0 and numteams-1

for (int i = 0; i < size; i++) {

array[i] = r.nextInt(numteams);

}

//Create threads, and later join them

for (int i = 0; i < numthreads; i++) {

threads[i] = new TeamStats(i);

threads[i].start();

}

for (int i = 0; i < numthreads; i++) {

threads[i].join();

}

//Combine and print team wins data

int mwins = 0;

int mteam = 0;

for (int i = 0; i < numteams; i++) {

int sumwins = 0;

for (int j = 0; j < numthreads; j++) {

sumwins += teamwins[j][i];

//Meanwhile, also record winning streak info to print later

if (threads[j].maxwins > mwins) {

mwins = threads[j].maxwins;

mteam = threads[j].maxteam;

}

}

System.out.println("Team " + i + " won " + sumwins + " games.");

}

System.out.println("Longest winning streak of " + mwins + " by team " + mteam);

}

@Override

public void run() {

int team = 0;

int numwins = 0;

//Process portion of array for each thread

for (int i = id \* set; i < (id + 1) \* set; i++) {

//For team wins, increment counter in teamwins array for this thread

//Cannot use one global array, because that could have race conditions

teamwins[id][array[i]]++;

//For winning streak, keep incrementing numwins if team is the same

//Then, if numwins > maxwins for this thread, update maxwins and maxteam;

if(array[i] == team) {

numwins++;

} else {

team = array[i];

numwins = 1;

}

//Check for wins by same team beyond current array portion

if(i + 1 == (id + 1) \* set) {

int j = i + 1;

while(j < size && array[j] == team) {

numwins++;

j++;

}

}

//Finally, update streak

updateStreak(numwins, team);

}

}

private void updateStreak(int numwins, int team) {

if(maxwins < numwins) {

maxwins = numwins;

maxteam = team;

}

}

}