/\*Resolucion enunciado\*/

public static final int MAXFILA = 7;

public static final int MAXCOLUMNA = 30;

public static void main(String[] args) {

//Doy por hecho que la matriz se precarga aqui

cargar\_matriz\_aleatorio\_secuencias\_int(matint);

realizarEjercicios(matInt);

}

private static void realizarEjercicios(int[][] matInt) {

recorrerMatriz(matInt);

}

private static void recorrerMatriz(int[][] matInt) {

for (int posFila = 0; posFila < MAXFILA; posFila++) {

obtenerSecuencias(matInt[posFila]);

}

}

private static void obtenerSecuencias(int[] mat) {

int inicio, fin;

inicio = 0;

fin = -1;

while ((inicio < MAXCOLUMNA - 1)) {

inicio = inicio\_secuencia(mat, fin + 1);

if (inicio < MAXCOLUMNA - 1) {

fin = fin\_secuencia(mat, inicio);

for (int pos = inicio; pos < fin; pos++) {

//\_esta ordenada de forma descendente y contiene el valor 6

if (es\_descendente(mat, inicio, fin) == true && pos == 6) {

corrimiento\_Izq(mat, inicio, fin);

} //\_tiene al menos 10 valores multiplos de 3,

else if (tieneValores(mat, inicio, fin) == true) {

corrimiento\_Izq(mat, inicio, fin);

} //\_no es capicua.

else if (esCapicua(mat, inicio, fin) == false) {

corrimiento\_Izq(mat, inicio, fin);

}

}

}

}

}

private static int inicio\_secuencia(int[] arr, int pos) {

while (pos < MAXCOLUMNA - 1 && arr[pos] == 0) {

pos++;

}

return pos;

}

private static int fin\_secuencia(int[] arr, int pos) {

while (pos < MAXCOLUMNA && arr[pos] != 0) {

pos++;

}

return (pos - 1);

}

public static void corrimiento\_Izq(int[] arr, int inicio, int fin) {

int pos = inicio;

while (pos < fin - 1) {

arr[pos] = arr[pos + 1];

pos++;

}

}

public static boolean es\_descendente(int[] arr, int inicio, int fin) {

int i = inicio;

while ((i < fin) && (arr[i + 1] < arr[i])) {

i++;

}

return (i == fin);

}

private static boolean tieneValores(int[] mat, int inicio, int fin) {

int i = inicio;

int contador = 0;

while ((i < fin) && (mat[i] % 3 == 0) && contador <= 10) {

i++;

contador++;

}

return (i == fin);

}

private static boolean esCapicua(int[] mat, int inicio, int fin) {

int i = inicio;

int j = fin;

while ((mat[i] == mat[j]) && (i != j)) {

i++;

j--;

}

return (i == j);

}