count = 0

//make 2D Boolean arr

boolean[][] visit = new boolean[r][c];

for (i=0, i<r; i++) {

boolean[] tempArr = new boolean[c];

for (j=0, j<c; j++) {

tempArr[j] = false;

}

visit[i] = tempArr;

}

//make 2D grid {

char[][] planet = new char[r][c];

for (i=0, i<r; i++) {

char[] tempArr = new char[c];

String[] plots = br.readline().split(“ “);

for (i=0, i<c; i++) {

tempArr[i] = plots[i];

}

planet[i] = tempArr;

}

}

//make Queue

Queue<Pair<Integer,Integer>> queue = new LinkedList<Pair<Integer,Integer>>;

for (i=0 to i=r) {

for (j=0 to j = c) {

type = planet[i][j]

pair cords = new pair(i,j);

if (type = L and visit[i][j] = false) {

count = count + 1;

queue.offer(coords);

visit[i][j] = true;

BFS(queue, r,c, visit)

}

}

}

BFS(queue,r,c,visit) {

while (!queue.empty()) {

currCell = queue.poll()

for (i=0 to 3) {

horizontalMove = dx[i]

vertMove = dy[i]

nextPosX = currCell.second() + horizontalMove;

nextPosY = currCell.first() + vertMove;

nextCell = new Pair(nextPosX,nextPosY)

if (0=<nexPosX<=c and 0<=nextPosY<=r and (planet[nextPosX][nextPosY] == L or C) and visit[nextPosX][nextPosY] = false ) {

visit[nextPosX][nextPosY] = true

queue.offer(nextCell)

}

}

}

}