***BOMBERMAN***

Bomberman is a popular game where a player is trapped in a room and is expected to get out by finding the key. In this version of the game, the room consists of breakable bricks, unbreakable walls and Villains. The player need to grab the key to win the game.

**Level 1:**

Create a NxN map where N is always an even number (Max Size = 26)

P - Player

\* - Wall

B - Brick

V - Villain

K - Key

-> First row and First column for location identification.

-> Walls position is always fixed.

***Input:***

Map Size

User position

Key position

No of Villains and their position

No of Bricks and their position

***Sample Input:***

Map size: 12

Player position: CB

Key position:FD

Villain count:2

V1: BH

V2: DF

Brick count:6

B1: DD

B2: ED

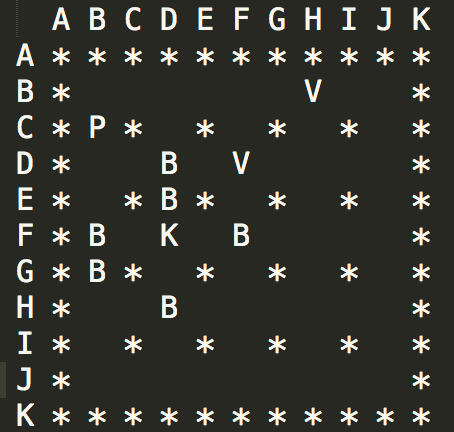
B3: FB

B4: FF

B5: GB

B6: HD

***Sample Output:***



**Note:** An element should not be placed on top of another element.

**Level 2:**

The Player can move in all 8 directions.

W - Move up

S - Move down

A - Move left

D - Move right

Q - Move diagonally up left

Z - Move diagonally down left

E - Move diagonally up right

C - Move diagonally down right

-> The player can not move to a position if there is a Wall or Brick

-> If there is a Villain where the player moves. The player dies

-> If there is a Key where the player moves. The player wins the game

-> The player can plant a bomb to destroy Bricks and Villains.

X - Bomb

***Map:***



***Sample Input:***

C

***Sample Output:***



-> Player position in above image is CD. Now the player plants a bomb in CD. Then moves to BC

***Sample Input:***

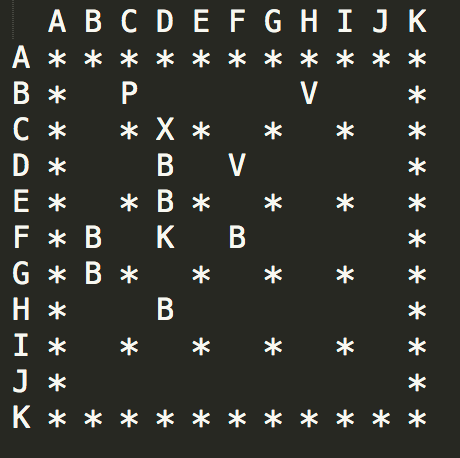
X

1 Plant

2 Detonate

1

***Sample Output:***



-> When the player moves out of the position the Bomb should display in the map. (Note: Again the player can not move on top of the Bomb)

-> The Bomb blast only in up, down, left, right directions and has the range of 1

-> If a Player plant a Bomb, then the player can not plant another Bomb until the first one detonates

***Sample Input:***

X

1 Plant

2 Detonate

2



***Note:*** -> If the player is inside the Bomb blast range. Then the player dies

-> Print the map each and every time after an input

***Level 3:***

-> In this Level the map will contain 3 Powers.

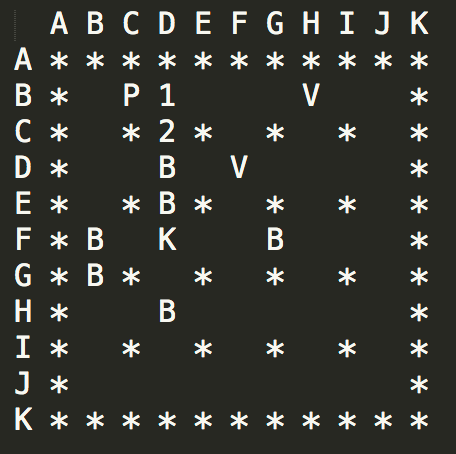
1 - Bomb Blast Range + 1

2 - Bomb can blast in diagonal direction

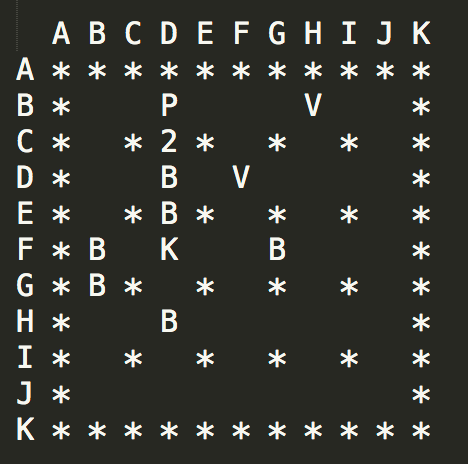
3 - Bomb count + 1

-> Get the position of powers 1,2 while creating the map.

***Sample Output:***



-> The Player moves to BD and gets the power 1



-> The Player moves to CD and gets the power 2



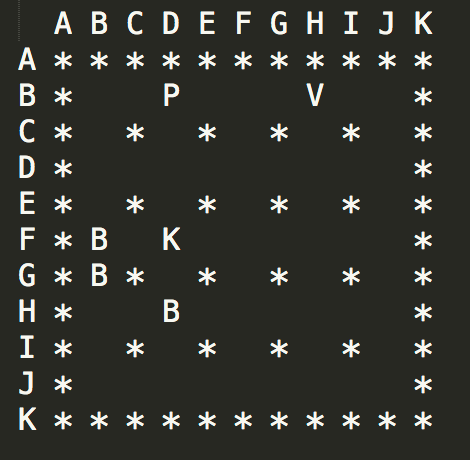
-> Then the Player moves to DE and plants a Bomb



-> Then the Player moves to CD and to BD

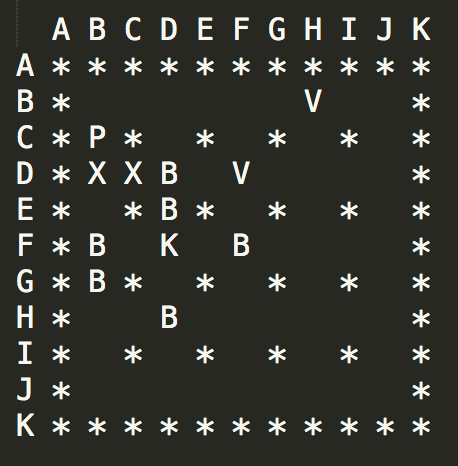


-> Now the Player detonates the Bomb



***Power 3 :***

-> If the player gets this power, the Bomb count increases by 1 (ie: The player can plant another Bomb while another Bomb already planted in the map)



-> If the Player got Power 3.

-> Ask whether to plant or detonate always.

-> As you can see the Player planted two bombs one in DB and another in DC

-> Now to detonate bomb Press 'X' and press '2' which blasts every bomb in the map.

-> If no bomb is planted and if the input is 2(Detonate) then print "No bomb is planted to detonate"

***Level 4:***

-> In this level, there will be a Dynamite which also denotes when it is triggered only by other blast

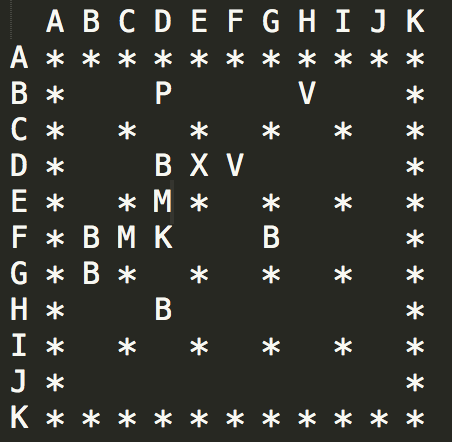
-> Dynamite always has the range of 1. And blasts in all directions

-> Get the position of Dynamite while creating the map

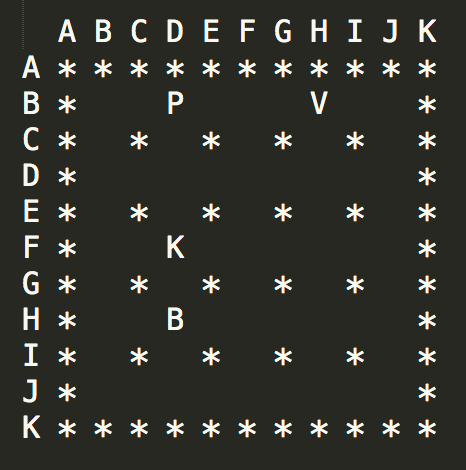
M - Dynamite

***Sample Output:***

-> Now Player P has power 1 and 2



-> Now the Player detonates the Bomb - 'X'

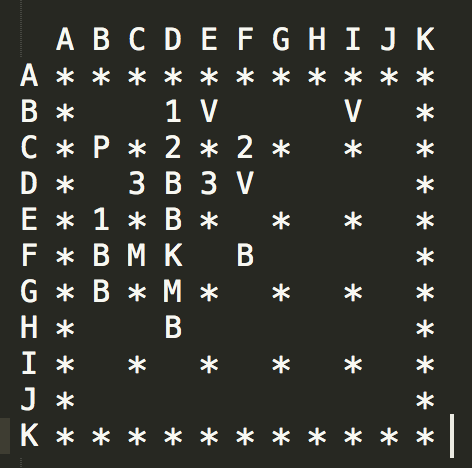


***Level 5:***

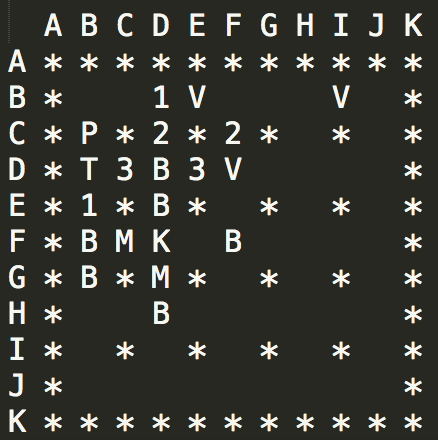
-> The player can plant a Timer Bomb.

T - Timer Bomb

Time = 3



-> Player moves to DB and places a bomb return back to CB



-> The Bomb detonates when the player moves 3 positions in any direction.

-> Timer Bomb always has the count - 1, range - 1, directions - (up, left, right, down)

***Note:*** The player can plant both the Ordinary bombs and Timer bomb at different position.

***Level 6:***

-> In this level, the Villains has also the ability to move in a direction

-> A Villain can move either in Horizontal or Vertical Direction

-> While creating the map get the direction of each villain

-> HRL - Horizontal Left

-> HRR - Horizontal Right

-> VRU - Vertical Up

-> VRD - Vertical Down

-> When Player moves one position in any direction. Then every Villain moves one position

-> If a Villain hits a Wall,Brick,Power,Key,Any Bomb then they moves in opposite direction

-> If Villains hits each other they die.

***Sample Input:***

Map size: 12

Player position: BC

Key position:FD

Villain count:3

V1: BH

Dir: HRL

V2: DF

Dir: VRD

Brick count:6

B1: DD

B2: ED

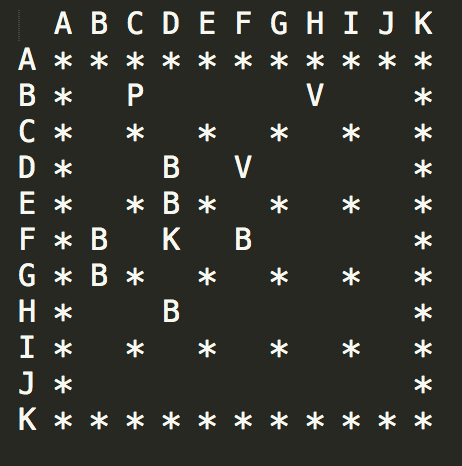
B3: FB

B4: FF

B5: GB

B6: HD

***Sample Output:***



-> Now Player moves to BD. Then each Villains moves one position



=====================================================================

**Test Cases:**

***Level 1:***

***T1***: To check whether the map prints correctly

mSize = 12; //Map Size

player = "CB"; //Player Location

key = "FD"; //Key Location

vCount = 3; // Villain Count

villain[0]="BI"; //Villain Location

villain[1]="DF"; //Villain Location

villain[2]="BE"; //Villain Location

bCount = 6; //Bricks Count

brick[0]="DD"; //Bricks Location

brick[1]="ED"; //Bricks Location

brick[2]="FB"; //Bricks Location

brick[3]="FF"; //Bricks Location

brick[4]="GB"; //Bricks Location

brick[5]="HD"; //Bricks Location

***T2***: To check whether any element on top of other element

mSize = 12;

player = "CB";

key = "CB";

vCount = 3;

villain[0]="BI";

villain[1]="DF";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

***T3***: To check whether any element on top of other element

mSize = 12;

player = "CB";

key = "FD";

vCount = 3;

villain[0]="BI";

villain[1]="DF";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FB";

***T4***: To check whether any element on top of other element

mSize = 12;

player = "AC";

key = "FD";

vCount = 3;

villain[0]="BI";

villain[1]="ED";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

***Level 2:***

***Input:***

mSize = 12;

player = "CB";

key = "FD";

vCount = 3;

villain[0]="BI";

villain[1]="DF";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

brick[4]="GB";

brick[5]="HD";

***T1:***

Move to EB. And Plant a bomb check whether brick alone destroys

***T2:***

Move to DE. And Plant a bomb check whether side brick and villain alone destroys

***T3:***

Move to FE. And Plant a bomb check whether right brick alone destroys

***T4:***

Plant a bomb near the player and detonate it. Check whether Player dies.

***T5:***

Check whether the player able to plant only one bomb

***Level 3:***

***Input:***

mSize = 12;

player = "CB";

key = "FD";

vCount = 3;

villain[0]="BI";

villain[1]="DF";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

brick[4]="GB";

brick[5]="HD";

rCount = 2; //Bomb Range + 1 Power

range[0]="BD"; //Power 1 Location

range[1]="EB"; //Power 1 Location

dCount = 2; //Diagonal Power

diag[0]="CD"; //Power 2 Location

diag[1]="CF"; //Power 2 Location

bombCount = 2; //Bomb Count + 1 Power

bombs[0]="DC"; //Power 3 Location

bombs[1]="DE"; //Power 3 Location

***T1:***

Move to EB get Power 1 and plant a bomb at EB and move to BB. Detonate the bomb and Check whether the bomb has blast range 2 which destroys the bomb at FB and GB

***T2:***

Move to EB get Power 1, Move to BD get Power1, Move to CD get Power 2 and Move to DE get Power 3. Then plant bomb at DE itself and Move to DB. Detonate the Bomb.

-> Destroyed places DD,ED,DF,GB

Note: Make sure you didn't get power 3 at location DC

***T3:***

-> Move to EB get Power 1, Move to DC get Power 3, Move to CD get Power 2, Move to BD get Power1, Move to DE get Power 3. Place a bomb one at DE and another at EB. Then move to BB to detonate the bombs.

-> Destroyed places DD,ED,DF,GB,FB,BE

***Level 4:***

***Input:***

mSize = 12;

player = "CB";

key = "FD";

vCount = 3;

villain[0]="BI";

villain[1]="DF";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

brick[4]="GB";

brick[5]="HD";

rCount = 2;

range[0]="BD";

range[1]="EB";

dCount = 2;

diag[0]="CD";

diag[1]="CF";

bombCount = 2;

bombs[0]="DC";

bombs[1]="DE";

dyCount = 2; //Dynamite Count

dynamo[0]="FC"; //Dynamite Location

dynamo[1]="GD"; //Dynamite Location

***T1:***

-> Move to BD get power 1, Move to CD get power 2, Move to DE get power 3. And place bomb in DE itself. Then Move to BB

-> Destroyed places DD,ED,DF,GB,FB,FC,GD,HD

***Level 5:***

***Input:***

mSize = 12;

player = "CB";

key = "FD";

vCount = 3;

villain[0]="BI";

villain[1]="DF";

villain[2]="BE";

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

brick[4]="GB";

brick[5]="HD";

rCount = 2;

range[0]="BD";

range[1]="EB";

dCount = 2;

diag[0]="CD";

diag[1]="CF";

bombCount = 2;

bombs[0]="DC";

bombs[1]="DE";

dyCount = 2;

dynamo[0]="FC";

dynamo[1]="GD";

***T1:***

-> Move to EB get Power 1, Move to DC get Power 3, Move to CD get Power 2, Move to BD get Power1, Move to DE get Power 3. Place a timer bomb at DE . Then move to BC to detonate the bomb (Which automatically blast).

-> Destroyed places DD,DF

***Level 6:***

***Input:***

mSize = 12;

player = "CB";

key = "FD";

vCount = 4

villain[0]="DH";

villainDir[0]="HRL"; //Villain Direction

villain[1]="EH";

villainDir[1]="VRU"; //Villain Direction

villain[2]="DG";

villainDir[2]="HRR"; //Villain Direction

villain[3]="CH";

villainDir[3]="VRD; //Villain Direction

bCount = 6;

brick[0]="DD";

brick[1]="ED";

brick[2]="FB";

brick[3]="FF";

brick[4]="GB";

brick[5]="HD";

rCount = 2;

range[0]="BD";

range[1]="EB";

dCount = 2;

diag[0]="CD";

diag[1]="CF";

bombCount = 2;

bombs[0]="DC";

bombs[1]="DE";

dyCount = 2;

dynamo[0]="FC";

dynamo[1]="GD";

***T1:***

Move to any location. Villain's destroyed location CH,DH,EH,DG

***T2:*** Change Villain's postion alone with below inputs

vCount = 4

villain[0]="DH";

villainDir[0]="HRL"; //Villain Direction

villain[1]="EH";

villainDir[1]="VRU"; //Villain Direction

villain[2]="DI";

villainDir[2]="HRR"; //Villain Direction

villain[3]="CH";

villainDir[3]="VRD; //Villain Direction

-> Move to any location. Villain's destroyed location CH,DH,EH,DI

***T3:*** Change Villain's postion alone with below inputs

vCount = 2

villain[0]="BE";

villainDir[0]="VRU"; //Villain Direction

villain[1]="BH";

villainDir[1]="HRR"; //Villain Direction

-> Move to any location. BE Villain can't move becasue no space up and down.

BH should move right. Check whether BH bounce back to opposite direction after hitting the wall. Then move the BH villain to BE Villain postion. Check whether both villain dies.

***T4:***

-> Chech whether Player dies when player and villain dashes at particular position.

**Program:**

package org.deep;

import java.util.NoSuchElementException;

import java.util.Scanner;

public class Map {

public static int[] val=new int[2];

public static int[] upos=new int[2];

public static int[] tpos=new int[2];

public static int[] dpos=new int[2];

public static int[][] bpos=new int[5][2];

public static int[][] vpos=new int[3][2];

public static int[][] map;

public static int[][] dv=new int[3][2];

public static int n;

public static int bomb=1;

public static int timer=3;

public static boolean isSet=false;

public static int planted=0;

public static int range=1;

public static boolean bf=false;

public static boolean isAlive=true;

public static boolean hasWon=false;

public static int nv;

public static String ip;

public static Scanner scan=new Scanner(System.in);

public static void dynamite(){

int i,j,rangei,no;

rangei=1;

no=2\*rangei+1;

try{

// System.out.println("In dynamite");

for(i=0-rangei;i<=rangei;i++){

for(j=0-rangei;j<=rangei;j++){

if(i==j||i==no/2-rangei||j==no/2-rangei||(i+j==0)){

if((dpos[0]+i)>=0&&(dpos[0]+i)<n&&(dpos[1]+j)>=0&&(dpos[1]+j)<n){

if(map[dpos[0]+i][dpos[1]+j]>=-1){

if(map[dpos[0]+i][dpos[1]+j]==5){

for(int k=0;k<nv;k++){

if(dpos[0]+i==vpos[k][0]&&dpos[1]+j==vpos[k][1]){

vpos[k][0]=-1;

vpos[k][1]=-1;

}

}

}

if(dpos[0]+i==upos[0]&&dpos[1]+j==upos[1]) isAlive=false;

map[dpos[0]+i][dpos[1]+j]=0;

}

}

}

}

}

}catch(ArrayIndexOutOfBoundsException e){

}

}

public static void detonate(){

try{

int i,j;

int no=2\*range+1;

while(planted>0){

planted--;

// System.out.println(dpos[0]+" "+dpos[1]);

if(bf){

for(i=0-range;i<=range;i++){

for(j=0-range;j<=range;j++){

if(i==j||i==no/2-range||j==no/2-range||(i+j==0)){

if((bpos[planted][0]+i)>=0&&(bpos[planted][0]+i)<n&&(bpos[planted][1]+j)>=0&&(bpos[planted][1]+j)<n){

if(map[bpos[planted][0]+i][bpos[planted][1]+j]>=-1){

if(map[bpos[planted][0]+i][bpos[planted][1]+j]==5){

for(int k=0;k<nv;k++){

if(bpos[planted][0]+i==vpos[k][0]&&bpos[planted][1]+j==vpos[k][1]){

vpos[k][0]=-1;

vpos[k][1]=-1;

}

}

}

if(bpos[planted][0]+i==upos[0]&&bpos[planted][1]+j==upos[1]) isAlive=false;

map[bpos[planted][0]+i][bpos[planted][1]+j]=0;

if(bpos[planted][0]+i==dpos[0]&&bpos[planted][1]+j==dpos[1]){

dynamite();

}

}

}

}

}

}

}

else{

for(i=0-range;i<=range;i++){

for(j=0-range;j<=range;j++){

if(i==no/2-range||j==no/2-range){

if((bpos[planted][0]+i)>=0&&(bpos[planted][0]+i)<n&&(bpos[planted][1]+j)>=0&&(bpos[planted][1]+j)<n){

if(map[bpos[planted][0]+i][bpos[planted][1]+j]>=-1){

if(map[bpos[planted][0]+i][bpos[planted][1]+j]==5){

for(int k=0;k<nv;k++){

if(bpos[planted][0]+i==vpos[k][0]&&bpos[planted][1]+j==vpos[k][1]){

vpos[k][0]=-1;

vpos[k][1]=-1;

}

}

}

if(bpos[planted][0]+i==upos[0]&&bpos[planted][1]+j==upos[1]) isAlive=false;

map[bpos[planted][0]+i][bpos[planted][1]+j]=0;

if(bpos[planted][0]+i==dpos[0]&&bpos[planted][1]+j==dpos[1]){

dynamite();

}

}

}

}

}

}

}

}

}catch(ArrayIndexOutOfBoundsException e){

}

}

public static void bombPlant(){

bpos[planted][0]=upos[0];

bpos[planted][1]=upos[1];

System.out.println(bpos[planted][0]+" "+bpos[planted][1]);

map[bpos[planted][0]][bpos[planted][1]]=7;

planted++;

}

public static void bombTime(){

try{

int i,j;

if(timer>0){

timer--;

}

else{

int no=2\*range+1;

for(i=0-range;i<=range;i++){

for(j=0-range;j<=range;j++){

if(i==no/2-range||j==no/2-range){

if((tpos[0]+i)>=0&&(tpos[0]+i)<n&&(tpos[1]+j)>=0&&(tpos[1]+j)<n){

if(map[tpos[0]+i][tpos[1]+j]>=-1){

if(map[tpos[0]+i][tpos[1]+j]==5){

for(int k=0;k<nv;k++){

if(tpos[0]+i==vpos[k][0]&&tpos[1]+j==vpos[k][1]){

vpos[k][0]=-1;

vpos[k][1]=-1;

}

}

}

if(tpos[0]+i==upos[0]&&tpos[1]+j==upos[1]) isAlive=false;

map[tpos[0]+i][tpos[1]+j]=0;

if(tpos[0]+i==dpos[0]&&tpos[1]+j==dpos[1]){

dynamite();

}

}

}

}

}

}

timer=3;

isSet=false;

}

}catch(ArrayIndexOutOfBoundsException e){

}

}

public static void plantTimerBomb(){

tpos[0]=upos[0];

tpos[1]=upos[1];

// System.out.println(tpos[0]+" "+tpos[1]);

map[tpos[0]][tpos[1]]=9;

isSet=true;

}

public static void dispMatrix(){

System.out.println(n);

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

for(int j=0;j<n;j++)

System.out.print(map[i][j]+" ");

System.out.println();

}

}

public static void mapInit(){

map=new int[n][n];

for(int i=0;i<n;i++)

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

if(i==0||j==0||i==n-1||j==n-1||((i%2==0)&&(j-i)%2==0))

map[i][j]=-2;

else map[i][j]=0;

}

}

public static void mapDisp(){

System.out.print(" ");

try{

for(int i=0;i<n;i++)

{

System.out.print((char)("A".charAt(0)+i));

System.out.print(" ");

}

System.out.println();

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

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

if(j==0) {

System.out.print((char)("A".charAt(0)+i));

System.out.print(" ");

}

switch(map[i][j]){

case 4:System.out.print("P");

break;

case 5:System.out.print("V");

break;

case 6:System.out.print("K");

break;

case 7:if(upos[0]==bpos[planted-1][0]&&upos[1]==bpos[planted-1][1])

System.out.print("P");

else System.out.print("X");

break;

case 8:System.out.print("M");

break;

case 9:if(upos[0]==tpos[0]&&upos[1]==tpos[1])

System.out.print("P");

else System.out.print("T");

break;

case 1:System.out.print("1");

break;

case 2:System.out.print("2");

break;

case 3:System.out.print("3");

break;

case -1:System.out.print("B");

break;

case -2:System.out.print("\*");

break;

default:System.out.print(" ");

break;

}

System.out.print(" ");

}

System.out.println();

}

}catch(ArrayIndexOutOfBoundsException e){

}

}

public static void value(){

try{

val[0]=(int)ip.charAt(0)-65;

val[1]=(int)ip.charAt(1)-65;

}

catch(StringIndexOutOfBoundsException e){

}

}

public static void mapCreate(){

// Scanner scan=new Scanner(System.in);

boolean flag=false;

if(flag){

System.out.println("Enter the Size of the Map: ");

n=scan.nextInt();

n--;

mapInit();

System.out.println("Enter the position of the user");

ip=scan.next();

value();

upos[0]=val[0];

upos[1]=val[1];

map[upos[0]][upos[1]]=4;

System.out.println("Enter the position of the key");

ip=scan.next();

value();

map[val[0]][val[1]]=6;

System.out.println("Enter the number of villains");

nv=scan.nextInt();

for(int i=0;i<nv;i++)

{

System.out.println("Enter coordinates for Villain " + (i+1));

ip=scan.next();

value();

vpos[i][0]=val[0];

vpos[i][1]=val[1];

map[val[0]][val[1]]=5;

System.out.println("Enter direction for villain "+(i+1));

ip=scan.next();

switch(ip){

case"HRL":dv[i][0]=0;dv[i][1]=-1;

break;

case"HRR":dv[i][0]=0;dv[i][1]=1;

break;

case"VRU":dv[i][0]=-1;dv[i][1]=0;

break;

case"VRD":dv[i][0]=1;dv[i][1]=0;

break;

default:System.out.println("Inalid direction\nDefault Set to HRR");

dv[i][0]=0;dv[i][1]=1;

break;

}

}

System.out.println();

System.out.println("Enter the number of bricks");

int bc=scan.nextInt();

for(int i=0;i<bc;i++)

{

System.out.println("Enter coordinates for Brick " + (i+1));

ip=scan.next();

value();

map[val[0]][val[1]]=-1;

}

System.out.println("Enter the coordinates for Power 1");

ip=scan.next();

value();

map[val[0]][val[1]]=1;

System.out.println("Enter the coordinates for Power 2");

ip=scan.next();

value();

map[val[0]][val[1]]=2;

System.out.println("Enter the coordinates for Power 3");

ip=scan.next();

value();

map[val[0]][val[1]]=3;

System.out.println("Enter the coordinates for Dyanamite");

ip=scan.next();

value();

dpos[0]=val[0];

dpos[1]=val[1];

map[val[0]][val[1]]=8;

}

else {

n=12;

n--;

mapInit();

ip="CB";

value();

upos[0]=val[0];

upos[1]=val[1];

map[upos[0]][upos[1]]=4;

ip="FG";

value();

map[val[0]][val[1]]=6;

nv=2;

ip="BH";

value();

vpos[0][0]=val[0];

vpos[0][1]=val[1];

map[val[0]][val[1]]=5;

ip="HRR";

switch(ip){

case"HRL":dv[1][0]=0;dv[1][1]=-1;

break;

case"HRR":dv[1][0]=0;dv[1][1]=1;

break;

case"VRU":dv[1][0]=-1;dv[1][1]=0;

break;

case"VRD":dv[1][0]=1;dv[1][1]=0;

break;

default:System.out.println("Enter Valid direction");

break;

}

ip="DF";

value();

vpos[1][0]=val[0];

vpos[1][1]=val[1];

map[val[0]][val[1]]=5;

ip="HRL";

switch(ip){

case"HRL":dv[0][0]=0;dv[0][1]=-1;

break;

case"HRR":dv[0][0]=0;dv[0][1]=1;

break;

case"VRU":dv[1][0]=-1;dv[1][1]=0;

break;

case"VRD":dv[1][0]=1;dv[1][1]=0;

break;

default:System.out.println("Enter Valid direction");

break;

}

ip="DD";

value();

map[val[0]][val[1]]=-1;

ip="ED";

value();

map[val[0]][val[1]]=-1;

ip="FB";

value();

map[val[0]][val[1]]=-1;

ip="FD";

value();

map[val[0]][val[1]]=-1;

ip="GB";

value();

map[val[0]][val[1]]=-1;

ip="HD";

value();

map[val[0]][val[1]]=-1;

ip="BC";

value();

map[val[0]][val[1]]=1;

ip="DC";

value();

map[val[0]][val[1]]=2;

ip="BI";

value();

map[val[0]][val[1]]=3;

ip="FC";

value();

dpos[0]=val[0];

dpos[1]=val[1];

map[val[0]][val[1]]=8;

}

// mapDisp();

}

public static int canMove(int mov[]){

int i=map[upos[0]+mov[0]][upos[1]+mov[1]];

if(i==0){

return 0;

}

else if(i>0&&i<7){

return i;

}

else return -1;

}

public static void move(int mov[]){

if(map[upos[0]][upos[1]]==4) map[upos[0]][upos[1]]=0;

upos[0]+=mov[0];

upos[1]+=mov[1];

map[upos[0]][upos[1]]=4;

}

public static void gameMove(){

try{

String ch="";

boolean flag1=false;

int choice;

int mov[]=new int[2];

System.out.println("Level 2!!\nEnter Choice");

if(ch.equals("")) ch=scan.next();

switch(ch){

case "W":mov[0]=-1;mov[1]=0;break;

case "E":mov[0]=-1;mov[1]=1;break;

case "D":mov[0]=0;mov[1]=1;break;

case "C":mov[0]=1;mov[1]=1;break;

case "Z":mov[0]=1;mov[1]=-1;break;

case "A":mov[0]=0;mov[1]=-1;break;

case "Q":mov[0]=-1;mov[1]=-1;break;

case "S":mov[0]=1;mov[1]=0;break;

case "X":flag1=true;break;

default:System.out.println("Invalid");

break;

}

if(flag1){

System.out.println("1:Plant\n2:Detonate");

choice=scan.nextInt();

if(choice==1){

System.out.println("1.Normal Bomb\n2.Timer Bomb");

int chc=scan.nextInt();

if(chc==1){

if(bomb-planted>0){

bombPlant();

}

else System.out.println("No more bombs to plant!!");

}

if(chc==2){

if(!isSet)

plantTimerBomb();

else System.out.println("Timer Bomb already planted");

}

}

else if(choice==2){

if(planted>0){

detonate();

}

else System.out.println("No bombs planted!!");

}

else System.out.println("Enter a valid choice!!");

}

else {

int num=canMove(mov);

if(num==0||num==4){

move(mov);

}

else if(num>0&&num<4){

addPower(num);

move(mov);

}

else if(num==5){

isAlive=false;

}

else if(num==6){

hasWon=true;

}

else {

System.out.println("Can't move there");

gameMove();

}

}

flag1=false;

ch="";

}

catch(NoSuchElementException e){

}

}

public static void addPower(int ch){

if(ch==1){

range+=1;

}

else if(ch==2){

bf=true;

}

else if(ch==3){

bomb+=1;

}

}

public static void villainMove(){

int i,num,j;

try{

for(i=0;i<nv;i++){

// System.out.println(vpos[i][0]+" "+vpos[i][1]);

if(vpos[i][0]==-1&&vpos[i][1]==-1) continue;

else{

// System.out.println(dv[i][0] + " " + dv[i][1]);

num=map[vpos[i][0]+dv[i][0]][vpos[i][1]+dv[i][1]];

if(num==-2||num==-1||num==9||num==7||num==6||num==3||num==2||num==1){

dv[i][0]=dv[i][0]\*-1;

dv[i][1]=dv[i][1]\*-1;

}

else if(num==4){

isAlive=false;

}

else if(num==5){

for(j=0;j<nv&&j!=i;j++){

if(vpos[j][0]==(vpos[i][0]+dv[i][0])&&vpos[j][1]==(vpos[i][1]+dv[i][1])){

map[vpos[j][0]][vpos[j][1]]=0;

vpos[j][0]=-1;

vpos[j][1]=-1;

}

}

// System.out.println("inhere "+vpos[i][0]+" "+vpos[i][1]);

map[vpos[i][0]][vpos[i][1]]=0;

vpos[i][0]=-1;

vpos[i][1]=-1;

break;

}

map[vpos[i][0]][vpos[i][1]]=0;

vpos[i][0]+=dv[i][0];

vpos[i][1]+=dv[i][1];

map[vpos[i][0]][vpos[i][1]]=5;

}

}

}catch(ArrayIndexOutOfBoundsException e){

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

mapCreate();//Level 1

// dispMatrix();

while(isAlive&&!hasWon){ //Entire game runs in this loop

if(isSet) bombTime();

if(isAlive){

mapDisp();

// System.out.println(bomb+ " "+range+" "+bf);

gameMove();

villainMove();

// dispMatrix();

}

// dispMatrix();

}

if(hasWon){

System.out.println("You Won!!!");

}

else System.out.println("You Lost!!!");

}

}