**2. The White Knight**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

class CodeChef {

public static void main(String[] args) throws NumberFormatException, IOException {

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

int T = Integer.parseInt(in.readLine());

for(int tests = 0; tests<T; tests++)

{

int N = Integer.parseInt(in.readLine());

char[][] board = new char[N][N];

int[][] intBoard = new int[N][N];

for(int line = 0; line<N; line++)

{

board[line] = in.readLine().toCharArray();

}

int currentValue = 0;

int u2r1 = 0;

int u1r2 = 0;

int d1r2 = 0;

int d2r1 = 0;

int answer = 0;

//start with the simplest case, the bottom right corner of the board.

for(int j = N-1; j>=0; j--)

{

for(int i=N-1; i>=0; i--)

{

u2r1 = 0;

u1r2 = 0;

d1r2 = 0;

d2r1 = 0;

currentValue =0;

if(i-2>=0 && j+1<N) //up two right one case

{

u2r1 = intBoard[i-2][j+1];

}

if(i-1>=0 && j+2<N) //up one right two case

{

u1r2 = intBoard[i-1][j+2];

}

if(i+1<N && j+2<N) //down one right two case

{

d1r2 = intBoard[i+1][j+2];

}

if(i+2<N && j+1<N) //down two right one case

{

d2r1 = intBoard[i+2][j+1];

}

currentValue = Math.max(Math.max(Math.max( u2r1, u1r2), d1r2), d2r1); //take max of all previously computed P spots

if(board[i][j]=='P')

{

currentValue += 1; //add 1 for current P

intBoard[i][j] = currentValue; //update intBoard for future Pawns

}

else if(board[i][j]=='K')

{

answer = currentValue; //if we are at the K, the answer is the currentValue, which is the max of any pawn path reached from there

break;

}

else

{

intBoard[i][j] = currentValue;

}

}

if(answer!=0)

break;

}

System.out.println(answer);

}

}

}