/\* Riya Gharat

N00901846

"Bug Jump"

Create an n by m array (n, m input by user). Initialize all cells in array to 0. The bug starts at cell 1,1. The bug can jump from its cell to any neighboring cell (left, right, up down, diagonal) – so there are at most 8 neighboring cells. Though, for example, cell 1,1 has only 3 neighboring cell, If the bug tries to jump off of the array, ignore that move. A move is determined by generating a random number between 1 and 8.

1 = left, 2 = right, 3 = up, 4 = down, 5 = up and right, 6 = up and left, 7 = down and right, 8 = down and left

Count how many moves it takes until the bug has visited each cell at least once. Display the array after each move, with a count for each cell of how many times the bug has visited that cell as well as what is the bug's current location.

\*/

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>

#define MAXSIZE 10

void createArray(int m, int arr[][m]);

void printArray(int m2, int arr2[][m2]);

int checkFull(int m3, int arr3[][m3]);

int main(void){

srand(time(NULL));

int col;

printf("Please enter a number of columns: ");

scanf("%d", &col);

int array[MAXSIZE][col];

// initializing array to 0

createArray(col, array);

// printing initial array

printArray(col, array);

printf("\n");

// Initial Position of Bug

int posRow = 1;

int posCol = 1;

int checkingFull = checkFull(col, array);

int count = 0;

while (checkingFull != 1){

int randomNum = (rand() % 8) + 1;

printf("Random Num = %d\n", randomNum);

if(randomNum == 1){

if((posCol - 1) >= 0){

posCol = posCol - 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 2){

if((posCol + 1) < col){

posCol = posCol + 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 3){

if((posRow - 1) >= 0){

posRow = posRow - 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 4){

if((posRow + 1) < MAXSIZE){

posRow = posRow + 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 5){

if(((posRow - 1) >= 0) && ((posCol + 1) < col)){

posRow = posRow - 1;

posCol = posCol + 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 6){

if(((posRow - 1) >= 0) && ((posCol - 1) >= 0)){

posRow = posRow - 1;

posCol = posCol - 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 7){

if(((posRow + 1) < MAXSIZE) && ((posCol + 1) < col)){

posRow = posRow + 1;

posCol = posCol + 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}else if(randomNum == 8){

if(((posRow + 1) < MAXSIZE) && ((posCol - 1) >= 0)){

posRow = posRow + 1;

posCol = posCol - 1;

array[posRow][posCol] ++;

count ++;

printArray(col, array);

printf("\n");

}

}

printf("Count = %d\n", count);

checkingFull = checkFull(col, array);

}

printf("\n");

printf("The number of times the bug moved: %d\n", count);

printf("The final position of the bug: %d, %d\n", (posRow+1),(posCol+1));

return 0;

}

// Creates the Array

void createArray(int m, int arr[][m]){

// creating an array with all the values set to zero

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

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

arr[i][j] = 0;

}

}

arr[1][1] = 1;

}

// Prints Array

void printArray(int m2, int arr2[][m2]){

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

for(int j = 0; j < m2; j++){

printf("%d ", arr2[i][j]);

}

printf("\n");

}

}

// Check if Array is Full

int checkFull(int m3, int arr3[][m3]){

int count = 0;

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

for(int j = 0; j < m3; j++){

if(arr3[i][j] == 0){

count = 1;

break;

}

}

}

if(count == 0)

return 1;

else

return 0;

}