```c
//
// Created by hengxin on 10/19/22.
// Run it with "Terminal"
//

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

#define SIZE 6

// extended_board as a parameter
// 1D array: (int, the address of the first element), the length
// 2D array: array of arrays [1][3]:
// address + 1 * (sizeof int) * size of col + (sizeof int) * 3
// 0: ------
// 1: ------
// 2: ------
void ExtendBoard(const int origin_board[][SIZE],
                 int extended_board[][SIZE + 2]);
void PrintExtendedBoard(const int extended_board[][SIZE + 2]);
void GenerateNewBoard(const int old_board[][SIZE + 2],
                      int new_board[][SIZE + 2]);
void CopyExtendedBoard(const int src_board[][SIZE + 2],
                       int dest_board[][SIZE + 2]);
void SleepAndClear(int sec);

int main() {
  const int board[SIZE][SIZE] = {
      { 0 },
      { 0, 1, 1, 0, 0, 0 },
      { 0, 1, 1, 0, 0, 0 },
      { 0, 0, 0, 1, 1, 0 },
      { 0, 0, 0, 1, 1, 0 },
      { 0 }
  };

  int old_board[SIZE + 2][SIZE + 2] = { 0 };
  ExtendBoard(board, old_board);
  PrintExtendedBoard(old_board);
//  SleepAndClear(1);

  int new_board[SIZE + 2][SIZE + 2] = { 0 };
  for (int round = 0; round < 10; round++) {
    GenerateNewBoard(old_board, new_board);
    SleepAndClear(1);
    PrintExtendedBoard(new_board);
    CopyExtendedBoard(new_board, old_board);
  }

  return 0;
}

```

```c
void ExtendBoard(const int origin_board[][SIZE],
                 int extended_board[][SIZE + 2]) {
  for (int row = 0; row < SIZE + 2; row++) {
    for (int col = 0; col < SIZE + 2; col++) {
      if (row == 0 || row == SIZE + 1 || col == 0 || col == SIZE + 1
) {
        extended_board[row][col] = 0;
      } else {
        extended_board[row][col] = origin_board[row - 1][col - 1];
      }
    }
  }
}

void PrintExtendedBoard(const int extended_board[][SIZE + 2]) {
  for (int row = 1; row <= SIZE; row++) {
    for (int col = 1; col <= SIZE; col++) {
      printf("%c ", extended_board[row][col] ? '*' : ' ');
    }
    printf("\n");
  }
}

void GenerateNewBoard(const int old_board[][SIZE + 2],
                      int new_board[][SIZE + 2]) {
  for (int row = 1; row <= SIZE; row++) {
    for (int col = 1; col <= SIZE; col++) {
      // count the number of neighbours of old_board[row][col]
      int neighbours =
          old_board[row - 1][col - 1] +
              old_board[row - 1][col] +
              old_board[row - 1][col + 1] +
              old_board[row][col - 1] +
              old_board[row][col + 1] +
              old_board[row + 1][col - 1] +
              old_board[row + 1][col] +
              old_board[row + 1][col + 1];

      // evaluate the new board
      if (old_board[row][col]) {  // old_board[row][col] is alive
        new_board[row][col] = (neighbours == 2 || neighbours == 3);
      } else {  // old_board[row][col] is dead
        new_board[row][col] = (neighbours == 3);
      }
    }
  }
}

void CopyExtendedBoard(const int src_board[][SIZE + 2],
                       int dest_board[][SIZE + 2]) {
  for (int row = 1; row <= SIZE; row++) {
    for (int col = 1; col <= SIZE; col++) {
      dest_board[row][col] = src_board[row][col];
    }
  }
}

void SleepAndClear(int sec) {
  sleep(sec);
  system("clear");
}
```