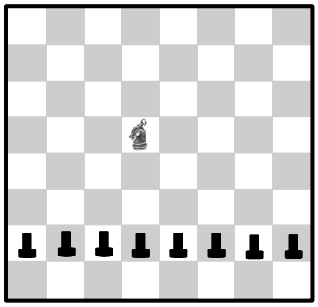
שאלה 6 (20 נק')

בתרגולים נלמדה בעיית הפרש: בהינתן נקודת התחלה על לוח שחמט NxN, עלינו למצוא מסלול של צעדי פרש המתחיל בנקודה זו ומבקר בכל משבצות הלוח פעם אחת בדיוק. הקוד לבעיה זו, כפי שנלמד בכיתה פרט לפונקציה `solve()`, מופיע בהמשך.
בשאלה זו נפתור וריאציה על בעיית הפרש. נתונים N חיילים אשר ניצבים בשורה האחת לפני האחרונה (השורה ה- N-2) ומוגדרים במערך גלובלי בשם `line` (ראה *שורה נטויה בקוד*). `line[i] == 0` אם ורק אם החייל בעמודה ה-i-ית עדיין **לא נאכל** על ידי הפרש (כאשר הפרש מגיע למשבצת בא מצוי חייל נאמר כי הפרש "אכל" את החייל).
הבעיה שעליכם לפתור (על ידי השלמת הפונקציה `solve_exam()`) היא כדלקמן: בהינתן נקודת התחלה על לוח שחמט NxN (הנח כי בתחילה הפרש אינו ממוקם במשבצת בה ניצב חייל), עליכם למצוא מסלול של צעדי פרש מנקודת ההתחלה כך שבמסלול זה הפרש "אוכל" את כל N החיילים, וזאת מבלי לעבור דרך אותה משבצת פעמיים. הפונקציה תחזיר 1 אם נמצא מסלול כזה ו-0 אחרת.
שימו לב: בשאלה זו על המסלול להסתיים כאשר החייל האחרון נאכל (כלומר כאשר החייל האחרון נאכל על הפונקציה להחזיר 1 באופן מיידי). בפרט, אין חובה על הפרש לבקר בכל משבצות המטריצה: מטריצת מהלכי הפרש צריכה להכיל אפסים במשבצות שבהן הפרש לא ביקר.

על הפתרון להיות בשיטת ה-backtracking.



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

#define N 8

int board[N][N] = {0};
int line[N] = {0};

int path_len = 0;

//==============================================================

int inboard(int row, int col)
{
   if (row>=0 && row<N && col>=0 && col<N)
       return 1;
   return 0;
}

//==============================================================

int legal_pos(int row, int col)
{
   return (inboard(row,col) && board[row][col]==0);
}
```

```
void get_legal_moves(int row, int col, int moves[8][2], int *n)
{
   *n = 0;
   if (legal_pos(row-2,col-1)){
       moves[*n][0]=row-2; moves[(*n)++][1]=col-1;}
   if (legal_pos(row-2,col+1)){
       moves[*n][0]=row-2; moves[(*n)++][1]=col+1;}
   if (legal_pos(row-1,col-2)){
       moves[*n][0]=row-1; moves[(*n)++][1]=col-2;}
   if (legal_pos(row-1,col+2)){
       moves[*n][0]=row-1; moves[(*n)++][1]=col+2;}
   if (legal_pos(row+1,col-2)){
       moves[*n][0]=row+1; moves[(*n)++][1]=col-2;}
   if (legal_pos(row+1,col+2)){
       moves[*n][0]=row+1; moves[(*n)++][1]=col+2;}
   if (legal_pos(row+2,col-1)){
       moves[*n][0]=row+2; moves[(*n)++][1]=col-1;}
   if (legal_pos(row+2,col+1)){
       moves[*n][0]=row+2; moves[(*n)++][1]=col+1;}
}

//==============================================================

void print_board()
{
   int i, j;
   for (i=0; i<N; ++i) {
     for (j=0; j<N; ++j)
       printf("%5d", board[i][j]);
     printf("\n\n");
   }
}

//==============================================================

// The function you have to implement
int solve_exam (int row, int col);

//==============================================================

int main()
{
   int begin_row = 0, begin_col = 0;

   board[begin_row][begin_col] = 1;
   path_len = 1;

   if (solve_exam(begin_row,begin_col)) {
     printf("\nsolution found for N=%d!\n\n", N);
     print_board();
   }
   else
     printf("no solution for N=%d.\n", N);
   return 0;
}
```

```
int solve_exam (int row, int col)
```