שאלה 4 (25 נקודות)

בשאלה זו נתכנת פונקציה שמנסה לסדר אבני דומינו על לוח. הלוח מתואר באמצעות מערך דו מימדי a שגודלו N על N (כש-N קבוע #define). חלק ממשבצות הלוח ריקות, וחלקן מלאות. הפונקציה תקבל את המערך הדו מימדי a כאשר בכל משבצת ריקה ישנו 0, ובכל משבצת מלאה יש 1-. עליכם לממש פונקציה שבודקת אם ניתן לרצף את משבצות הלוח הריקות בעזרת אבני דומינו בגודל 1על 2, ובמידה וכן, מחזירה את הפתרון שמצאה.

אבני הדומינו יצוינו על ידי מספרים טבעיים: ...1,2,3 (ניתן להניח שמספר אבני הדומינו אינו מוגבל). בתום ריצת הפונקציה, אם אכן ניתן לרצף את הלוח אזי התא `a[i][j]` יכיל את מספרה של אבן הדומינו שמוקמה בתא זה (אם לא ניתן לרצף אז אין חשיבות לתוכן a בסוף הריצה). כמו כן, הפונקציה תחזיר 1 במידה ומצאה פתרון, ו-0 אם אין פתרון. שימו לב שניתן למקם אבן דומינו על הלוח במאוזן או במאונך – ראו דוגמאות להלן.

הערה: אם יש יותר מאפשרות אחת לריצוף הלוח, ניתן להחזיר פתרון אחד כלשהו.

<u>דוגמא ראשונה:</u>

N=3

```
a[3][3]={  0,    0,    0
           0,    -1,   0,
           0,    0,    0 }
```

אחרי ריצת הפונקציה פתרון אפשרי הוא:

```
a[3][3]={  1,    1,    2
           3,    -1,   2,
           3,    4,    4 }
```

<u>דוגמא שנייה:</u>

N=3

```
a[3][3]={  0,    -1,   0
           0,    0,    0,
           0,    0,    0 }
```

לוח כזה אי אפשר לרצף.

לנוחותכם, ניתן להשתמש בפונקציות העזר הבאות:

פונקציה שמחזירה 1 אם התא (i, j) בתחום הלוח:

```
int on_board(int i, int j)
{
      if (i>=0 && i<N && j>=0 && j<N)
            return 1;
      return 0;
}
```

פונקציה שמקבלת זוג אינדקסים (i, j) של ריבוע בלוח, ומעדכנת אותם כך שיכילו את האינדקסים של הריבוע הבא בלוח. הפונקציה מתקדמת לריבוע הבא באותה השורה במידה ויש כזה, או לריבוע הראשון בשורה הבאה במידה והגענו לקצה השורה. הפונקציה מחזירה 0 אם הגענו לקצה הלוח:

```
int next_square(int* i, int* j)
{
      (*j)++;
      if (*j == N)
      {
            *j = 0;
            (*i)++;
            if (*i == N)
                  return 0;
      }
      return 1;
}
```

פונקציה שמחזירה 1 אם התא (i, j) הינו ריבוע פנוי בלוח (0 אחרת):

```
int square_is_free(int i, int j, int a[N][N])
{
      if (on_board(i,j) && a[i][j]==0)
            return 1;
      return 0;
}
```

פונקציה שמחזירה 1 אם הלוח מלא (0 אחרת):

```
int board_is_full(int a[N][N])
{
      int i=0, j=0;
      do {
            if (square_is_free(i,j,a))
                  return 0;
      } while (next_square(&i, &j);

      return 1;
}
```

פונקציה שמחזירה 1 אם ניתן להניח לוחית דומינו בתא ה-(i, j) בלוח, ו-0 אחרת. עבור הפרמטר horizontal==1 ייבדקו זוג הריבועים האופקיים (i, j), (i, j+1), ואילו עבור הפרמטר horizontal==0 ייבדקו זוג הריבועים האנכיים (i, j) (i+1, j):

```
int can_put_tile(int i, int j, int a[N][N], int horizontal)
{
      if (!square_is_free(i,j,a))
            return 0;
      if (horizontal && square_is_free(i,j+1,a))
            return 1;
      if (!horizontal && square_is_free(i+1,j,a))
            return 1;
      return 0;
}
```

נתונה פונקצית המעטפת הבאה, המקבלת את הלוח ההתחלתי כפרמטר, וכותבת לתוכו את הפתרון (אם קיים כזה). הפונקציה מחזירה 1 אם אמנם נמצא פתרון, ו-0 אחרת. שימו לב שפונקציה זו עוטפת קריאה לפונקציה **tile_board_aux**. עליכם להשלים את הפרמטרים המתאימים בקריאה לפונקציה **tile_board_aux**, ולממש פונקציה זו בדף הבא.

שימו לב: ניתן לקבוע את זהות ומספר הפרמטרים של הפונקציה **tile_board_aux** כרצונכם.

```
int tile_board(int a[N][N])
{
    return tile_board_aux(a, 1);
}
```

```
int tile_board_aux( int a[N][N], int index ) {

  int i=0, j=0;

  if (board_is_full(a))
    return 1;

  // check for all i,j on the board if tile can be placed
  do {
    if (can_put_tile(i,j,a,0)) {    // vertical positioning
      a[i][j] = index;
      a[i+1][j] = index;
      if (tile_board_aux(a, index+1))
        return 1;
      a[i][j] = 0;
      a[i+1][j] = 0;
    }

    if (can_put_tile(i,j,a,1)) {   // horizontal positioning
      a[i][j+1] = index;
      a[i][j] = index;
      if (tile_board_aux(a, index+1))
        return 1;
      a[i][j+1] = 0;
      a[i][j] = 0;
    }

  } while (next_square(&i, &j));

  return 0;
}
```