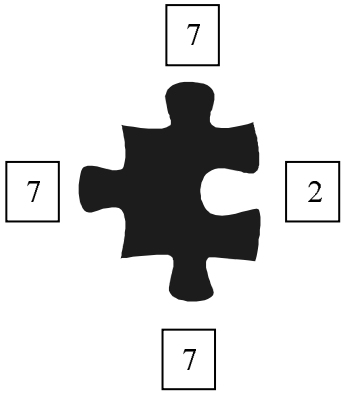
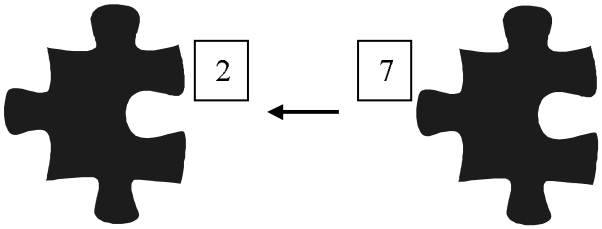
שאלה 4 (25 נקודות)

בשאלה זו נכתוב תוכנית שמרכיבה פאזל. אנו נניח כי הפאזל מורכב מ- N^2 חתיכות, אותן יש למקם זו לצד זו על פני לוח בגודל NxN. כל חתיכה מהפאזל היא ריבועית ובעלת 4 צלעות, ולכל צלע יש אחת מ-10 צורות חיבור, הממוספרות 0..9. לדוגמה, לחתיכת הפאזל הבאה יש צד ימני עם חיבור מסוג 2, ושלושה צדדים נוספים בעלי חיבור מסוג 7:



על מנת להרכיב את הפאזל, על חתיכות הפאזל השונות להתאים זו לזו בצורת החיבור שלהן. הכלל לחיבור הוא פשוט: חיבור מסוג X מסוגל להתחבר אך ורק לחיבור מסוג (X–9). למשל, עבור החתיכה שלמעלה, הצלע הימנית (חיבור מסוג 2) יכולה להתחבר לצלע השמאלית של חתיכה דומה (חיבור מסוג 7):



אנו נניח, לשם הפשטות, כי כל חתיכות הפאזל מסובבות מראש לכיוון הסופי שלהן בפאזל; כלומר, אין צורך לנסות ולסובב כל חתיכה כאשר מניחים אותה על הלוח. שימו לב שעל מנת למקם חתיכה על הלוח, הצלע הימנית שלה צריכה להתאים לצלע השמאלית של החתיכה שמימינה, הצלע העליונה שלה צריכה להתאים לצלע התחתונה של החתיכה שמעליה, וכן הלאה. עבור חתיכה שנמצאת בקצה הלוח, כל סוג של חיבור יכול להוות שפה של הפאזל (כלומר אין חתיכות מיוחדות שהן פינות או שפות, אלא כל חתיכה יכולה להיות בקצה הלוח).

חלקי הפאזל נתונים במערך דו-ממדי p[N*N][4], כש-N הוא קבוע המוגדר כ-#define. השורה ה-i במערך הדו-מימדי מתאימה לחתיכת הפאזל ה-i, כאשר p[i][0] מכיל את צורת החיבור של הצלע הימנית, p[i][1] את צורת החיבור של הצלע השמאלית, p[i][2] של הצלע העליונה ו-p[i][3] של הצלע התחתונה. לנוחותכם, הוגדרו הקבועים הבאים המייצגים את האינדקסים של 4 הצלעות:

```
#define RIGHT 0
#define LEFT 1
#define TOP 2
#define BOTTOM 3
```

עליכם לממש את הפונקציה solvepuzzle (בדף הבא), המקבלת כקלט את מערך חלקי הפאזל p[N*N][4], ומחזירה 1 במידה ויש פתרון חוקי לפאזל ו-0 אחרת. הפונקציה מקבלת גם מערך פלט דו-ממדי board[N][N] המייצג את הלוח. במידה ויש פתרון, יש לכתוב למערך זה את הסידור של חלקי הפאזל על הלוח – בכל תא יש לכתוב את האינדקס של חתיכת הפאזל שנמצאת בתא זה. במידה ואין פתרון, אין חשיבות לתוכן הלוח בתום הריצה. שימו לב שהמערך board אינו בהכרח מאותחל לתוכן כלשהו בעת הקריאה לפונקציה.

לנוחותכם, ניתן בפתרון להשתמש בפונקצית העזר הבאה. פונקציה זו מקבלת את הלוח board, את מערך החלקים p, אינדקסים i,j של תא בלוח ואינדקס id של אחת מחתיכות הפאזל. הפונקציה מחזירה 1 במידה וניתן למקם את חתיכת הפאזל בתא i,j, ו-0 אחרת. **שימו לב** שפונקציה זו מניחה שתאים ריקים בלוח מכילים את הערך 1-.

```
int islegal(int board[N][N], int p[N*N][4],
            int i, int j, int id)
{
  if (board[i][j] != -1)
    return 0;

  // upper piece
  if (i>0 && board[i-1][j] != -1 &&
      p[board[i-1][j]][BOTTOM] != 9-p[id][TOP])
    return 0;

  // lower piece
  if (i<N-1 && board[i+1][j] != -1 &&
      p[board[i+1][j]][TOP] != 9-p[id][BOTTOM])
    return 0;

  // left piece
  if (j>0 && board[i][j-1] != -1 &&
      p[board[i][j-1]][RIGHT] != 9-p[id][LEFT])
    return 0;

  // right piece
  if (j<N-1 && board[i][j+1] != -1 &&
      p[board[i][j+1]][LEFT] != 9-p[id][RIGHT])
    return 0;

  return 1;
}
```

```
int solvepuzzle(int board[N][N], int p[N*N][4])
{
  int i, j;
  for (i=0; i<N; ++i)
    for (j=0; j<N; ++j)
      board[i][j] = -1;

  return solvepuzzle_aux(board,p,0);
}


int solvepuzzle_aux(int board[N][N], int p[N*N][4], int id)
{
  int i, j;

  if (id==N*N)
    return 1;

  for (i=0; i<N; ++i) {
    for (j=0; j<N; ++j)
    {
      if (islegal(board,p,i,j,id))
      {
        board[i][j] = id;
        if(solvepuzzle_aux(board,p,id+1))
          return 1;
        board[i][j] = -1;
      }
    }
  }
  return 0;
}
```

הסבר: נפתור את הבעיה באמצעות backtracking. הפרמטר id בפונקצית העזר מציין את האינדקס של חתיכת הפאזל הנוכחית שאנו מנסים למקם. כעת, בכל קריאה רקורסיבית נעבור על כל האפשרויות למיקום חתיכת הפאזל הנוכחית על הלוח, ולכל מיקום חוקי נשים את החתיכה על הלוח ונבדוק רקורסיבית אם ניתן להרחיב את הפתרון שקיבלנו לפתרון מלא. במידה וכן – נחזיר 1, ואחרת (במידה ואף אפשרות למיקום החתיכה לא הצליחה) נחזיר 0. בפונקצית המעטפת אנו מאתחלים את הלוח לערכי (1-) שמציינים משבצות ריקות בלוח, ומבצעים קריאה לפונקצית העזר הרקורסיבית.