Matrice

1. Data je šahovska tabla na kojoj je raspoređeno osam dama. Napišite program koji proverava da li se neke dve dame napadaju (dve dame se napadaju ako se nalaze u istoj vrsti, istoj koloni ili na istoj dijagonali). Sa standardnog ulaza učitava se 0-1 matrica dimenzije 8x8 čijih 8 jedinica opisuje položaj 8 dama. Na standardnom izlazu ispisati tekst `NE` ako se dame ne napadaju ili `DA` ako se neke dve dame napadaju.
Primer 1
Ulaz
0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
Izlaz
NE

Primer 2
Ulaz
0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0
Izlaz
DA

Resenje 01
IDEJA 1: Proverimo svaku vrstu, svaku kolonu i svaku dijagonalu, i izbrojmo koliko se na njima nalazi dama. Čim ustanovimo da je broj dama veći od 1, brojanje možemo prekinuti i možemo ustanoviti da se dame napadaju. Dame ćemo čuvati u matrici logičkih vrednosti dimenzije 8x 8.

```
#include <iostream>
#include <algorithm>

using namespace std;
```

```
int tabla[8][8];

void ucitaj() {
  for (int v = 0; v < 8; v++)
    for (int k = 0; k < 8; k++)
      cin >> tabla[v][k];
}

bool proveraKolona() {
  for (int k = 0; k < 8; k++) {
    int brojDama = 0;
    for (int v = 0; v < 8; v++)
      if (tabla[v][k]) {
          brojDama++;
          if (brojDama > 1)
            return true;
      }
  }
  return false;
}

bool proveraVrsta() {
  for (int v = 0; v < 8; v++) {
    int brojDama = 0;
    for (int k = 0; k < 8; k++)
      if (tabla[v][k]) {
          brojDama++;
          if (brojDama > 1)
            return true;
      }
  }
  return false;
}

bool proveraGlavnihDijagonala() {
  for (int d = -7; d <= 7; d++) {
    int brojDama = 0;
    for (int v = max(0, -d); v < min(8, 8-d); v++)
      if (tabla[v][v+d]) {
          brojDama++;
          if (brojDama > 1)
            return true;
      }
  }
```

```
  return false;
}

bool proveraSporednihDijagonala() {
  for (int d = 0; d <= 14; d++) {
    int brojDama = 0;
    for (int v = max(0, d-7); v < min(8, d+1); v++)
      if (tabla[v][d-v]) {
          brojDama++;
          if (brojDama > 1)
            return true;
      }
  }
  return false;
}

int main() {
  ucitaj();
  if (proveraVrsta() ||
      proveraKolona() ||
      proveraGlavnihDijagonala() ||
      proveraSporednihDijagonala())
    cout << "DA" << endl;
  else
    cout << "NE" << endl;
  return 0;
}
```

VAŽNO: U glavnom programu proveravamo da li je neka od provera vratila `true`. Recimo i da se, zbog lenjog izračunavanja operatora `||` kojim su četiri poziva funkcija povezana, ako neka od funkcija vrati `true` one nakon nje ni ne pozivaju.

Resenje 02

Drugi način da rešimo zadatak je da proverimo sve parove dama i da za svake dve dame proverimo da li se nalaze na istoj vrsti, istoj koloni ili na istoj dijagonali. Položaj dama možemo zabeležiti u matrici celih brojeva dimenzije 8 x 2 tako što ćemo u svakoj vrsti te matrice čuvati broj vrste i broj kolone u kojoj se nalazi neka dama.

Dva polja (v1, k1) i (v2, k2) se nalaze
u istoj vrsti ako je v1 = v2,
u istoj koloni ako je k1 = k2,
na istoj dijagonali ako je |v1 - v2| = |k1 - k2| (tj. ako je horizontalno rastojanje ta dva polja jednako vertikalnom).

```
#include <iostream>
#include <cmath>

using namespace std;

int dame[8][2];

void ucitaj() {
  int d = 0;
  for (int v = 0; v < 8; v++)
    for (int k = 0; k < 8; k++) {
      int p;
      cin >> p;
      if (p == 1) {
          dame[d][0] = v;
          dame[d][1] = k;
          d++;
      }
    }
}

bool proveraParovaDama() {
  for (int d1 = 0; d1 < 8; d1++)
    for (int d2 = d1+1; d2 < 8; d2++) {
      if (dame[d1][0] == dame[d2][0] ||
            dame[d1][1] == dame[d2][1] ||
            abs(dame[d1][0] - dame[d2][0]) == abs(dame[d1][1] - dame[d2][1]))
          return false;
    }
  return true;
}

int main() {
  ucitaj();
  if (proveraParovaDama())
    cout << "NE" << endl;
  else
    cout << "DA" << endl;
  return 0;
}
```

2. Napisati program koji sa standardnog ulaza učitava kvadratnu matricu kojom je predstavljena neka relacija i ispisuje da li relacija je refleksivna, simetrična, tranzitivna. Dakle, elementi matrice su 0 i 1 (vrednost 0 u i-tom redu i j-toj koloni

označava da i-ti element skupa nije u relaciji sa j-tim elementom). U prvoj liniji standardnog ulaza nalazi se dimenzija matrice $n < 20$, a zatim se učitava n redova sa po n elemenata. Elementi matrice su razdvojeni blanko znakom.

ULAZ
4
1 0 0 0
0 1 1 0
0 0 1 0
0 0 0 0

IZLAZ
Relacija nije refleksivna.
Relacija nije simetricna.
Relacija jeste tranzitivna.

*3. U odnosu na prethodni zadatak
a) napisati funkciju koja određuje refleksivno zatvorenje relacije (najmanju refleksivnu relaciju koja je nadskup date).
b) Napisati funkciju koja određuje simetrično zatvorenje relacije (najmanju simetričnu relaciju koja je nadskup date). (
c) Napisati funkciju koja određuje refleksivno-tranzitivno zatvorenje relacije (najmanju refleksivnu i tranzitivnu relaciju koja sadrži datu).
Napomena: Koristiti Varšalov algoritam za minimalno tranzitivno zatvorenje

ULAZ
4
1 0 0 0
0 1 1 0
0 0 1 0
0 0 0 0

IZLAZ
Refleksivno zatvorenje relacije:
1 0 0 0
0 1 1 0
0 0 1 0
0 0 0 1
Simetricno zatvorenje relacije:
1 0 0 0
0 1 1 0
0 1 1 0
0 0 0 0
Refleksivno-tranzitivno zatvorenje relacije:

1 0 0 0
0 1 1 0
0 0 1 0
0 0 0 1

Resenje
c)

```
/* Funkcija odredjuje refleksivno-tranzitivno zatvorenje zadate
   relacije koriscenjem Varsalovog algoritma */
void ref_tran_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
{
  int i, j, k;

  /* Odredjuje se refleksivno zatvorenje matrice */
  ref_zatvorenje(m, n, zatvorenje);

  /* Primenom Varsalovog algoritma odredjuje se tranzitivno
     zatvorenje matrice */
  for (k = 0; k < n; k++)
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
        if ((zatvorenje[i][k] == 1) && (zatvorenje[k][j] == 1)
            && (zatvorenje[i][j] == 0))
          zatvorenje[i][j] = 1;
}
```

4. Data je kvadratna matrica dimenzije n×n.
(a) Napisati funkciju koja određuje najveći element matrice na sporednoj dijagonali.
(b) Napisati funkciju koja određuje indeks kolone koja sadrži najmanji element matrice.
(c) Napisati funkciju koja određuje indeks vrste koja sadrži najveći element matrice.
(d) Napisati funkciju koja određuje broj negativnih elemenata matrice.
Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati u prvom redu broj vrsta n ($0 < n \leq 32$) celobrojne kvadratne matrice, a potom i svaku vrstu u posebnom redu. Na standardni izlaz ispisati rezultat primene prethodno napisanih funkcija.

ULAZ
3
1 2 3
 -4 -5 -6
7 8 9

IZLAZ
Najveci element sporedne dijagonale je 7.
Indeks kolone sa najmanjim elementom je 2.
Indeks vrste sa najvecim elementom je 2.
Broj negativnih elemenata matrice je 3.

Resenje

```
#include <iostream>

using namespace std;

const int  NMAX=32;

/* Funkcija izracunava najveci element na sporednoj dijagonali. Za
   elemente sporedne dijagonale vazi da je zbir indeksa vrste i
   indeksa kolone jednak n-1 */
int max_sporedna_dijagonala(int m[][NMAX], int n)
{
    int i;
    int max_na_sporednoj_dijagonali = m[0][n - 1];

    for (i = 1; i < n; i++)
        if (m[i][n - 1 - i] > max_na_sporednoj_dijagonali)
            max_na_sporednoj_dijagonali = m[i][n - 1 - i];

    return max_na_sporednoj_dijagonali;
}

/* Funkcija izracunava indeks kolone najmanjeg elementa */
int indeks_min(int m[][NMAX], int n)
{
    int i, j;
    int minimum = m[0][0], indeks_kolone = 0;

    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            if (m[i][j] < minimum)
            {
                minimum = m[i][j];
                indeks_kolone = j;
            }

    return indeks_kolone;
```

```
}

/* Funkcija izracunava indeks vrste najveceg elementa */
int indeks_max(int m[][NMAX], int n)
{
   int i, j;
   int maximum = m[0][0], indeks_vrste = 0;

   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         if (m[i][j] > maximum)
         {
            maximum = m[i][j];
            indeks_vrste = i;
         }
   return indeks_vrste;
}

/* Funkcija izracunava broj negativnih elemenata matrice */
int broj_negativnih(int m[][NMAX], int n)
{
   int i, j;
   int broj_negativnih = 0;

   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         if (m[i][j] < 0)
            broj_negativnih++;

   return broj_negativnih;
}

int main()
{
   int m[NMAX][NMAX];
   int n;
   int i, j;

   /* Ucitava se broj vrsta matrice */
   cin >> n;


   /*if (n > NMAX || n <= 0) {
     cout << "Greska: Neodgovarajuci broj vrsta matrice.\n" << endl;
     return 0;
```

```
    }*/

    /* Ucitava se matrica */
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            cin >> m[i][j];

    /* Ispisuju se rezultati izracunavanja */
    cout << "Najveci element sporedne dijagonale je " <<
max_sporedna_dijagonala(m, n)  << endl;

    cout << "Indeks kolone sa najmanjim elementom je " << indeks_min(m, n) <<
endl;

    cout << "Indeks vrste sa najvecim elementom je " << indeks_max(m, n) << endl;

    cout <<"Broj negativnih elemenata matrice je " << broj_negativnih(m, n) << endl;

    return 0;
}
```