

CATEDRA

FIZICĂ, MATEMATICĂ ȘI TEHNOLOGII INFORMAȚIONALE

SPECIALITATEA

TEHNOLOGII INFORMAȚIONALE ÎN INSTRUIRE

Structuri algebrice pe calculator

Laboratorul Nr.7

Realizat: Cojucovschi Ion

Grupa: CIII

Verificat: Chiriac Liubomir

Chișinău, 2018

Algoritmi privind verificarea si obținerea izomorfismelor de grupoizi

Problemă.Fie că avem doi grupoizi ()și () de același ordin n . Să se verifice dacă () este izomorf cu (). Algoritmul de verificare al grupoizilor la izomorfism este următorul:

1. Se introduce dimensiunea n a grupoizilor.
2. Se introduce grupoidul () din n elemente.
3. Se introduce grupoidul () din n elemente.
4. Se generează toate substituțiile și se testează condiția:
5. Se afișează la monitor acele substituții pentru care obținem izomorfism.
6. Se afișează la monitor rezultatul obținut: grupoizii dați (), () sunt izomorfi pentru substituțiileori grupoizii respectivi nu sînt izomorfi.

Pentru fiecare din exemple, să se verifice dacă există izomorfism între () și ().

Ex.1					Ex.2					Ex.3				Ex.4			
·	1	2	3	4	·	1	2	3	4	·	1	2	3	·	1	2	3
1	1	2	3	4	1	1	2	3	4	1	1	2	3	1	1	2	3
2	2	3	4	1	2	2	1	4	3	2	3	1	2	2	2	3	1
3	4	1	2	3	3	3	4	1	2	3	2	3	1	3	3	1	2
4	3	4	1	2	4	4	3	2	1								
*	1	2	3	4	*	1	2	3	4	*	1	2	3	*	1	2	3
1	1	2	3	4	1	2	1	4	3	1	3	2	1	1	3	1	2
2	2	4	1	3	2	1	2	3	4	2	1	3	2	2	2	3	1
3	4	3	2	1	3	4	3	2	1	3	2	1	3	3	1	2	3
4	3	1	4	2	4	3	4	1	2								

Elaborarea programului:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Laboratorul_numarul_7
{
    class Program
```

```

{
    static int r1, d1, r, d, r2, n, t, i, j, k, p, d2;
    static int[] st = new int[25];
    static string f3;
    static string f1;
    static string f2;
    static int[,] gr, m, m1, m2, m3, m4 = new int[25, 25];

```

```

static void Main()
{
    int[,] a = new int[3, 3]    { {0,0,0 },
                                { 0,1,2},
                                { 0,2,1} };

    int n = 0;

    Console.WriteLine("Dati ordinul matricei");
    n = Convert.ToInt32(Console.ReadLine());
    r = 0; r1 = 0;

```

```

    afisare(a, n);
    asociativ(a, n);
    medial(a, n);
    paramedial(a, n);
    bicomutativ(a, n);
    ag_gr(a, n);
    ga_gr(a, n);
    ga_gr1(a, n);
    ad_gr(a, n);
    da_gr(a, n);
    hexagonal(a, n);
    dist_dr(a, n);
    dist_st(a, n);
    unitate_dreapta(a, n, out r);
    unitate_stanga(a, n, out r1);
    unitate(ref r, ref r1);
    ward(a, n);
    ward_invers(a, n);

```

```

    Console.ReadKey();

```

```

}

```

```

public void initializeF3(int n)
{
    bool iz = false;
    int level = fact(n);
    for (int k = 1; k < level + 1; k++)
    {
        for (int i = 1; i < n; i++)
        {

        }
    }
}

```

```

}

```

```

public int fact(int n)
{

```

```

    int val = 1;
    for (int i = 1; i < n; i++)
        val = val * i;
    return val;
}

```

```

public void initializeF2()
{
    f2 += "(Q,.) \n";
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 0; j < n + 1; j++)
        {
            f2 += m[i, j] + " ";
        }
        f2 += "\n";
    }
}

```

```

f2 += "\n\n(Q,*) \n";
for (int i = 1; i < n + 1; i++)
{
    for (int j = 0; j < n + 1; j++)
    {
        f2 += m4[i, j] + " ";
    }
    f2 += "\n";
}

```

```

}

```

```

public void backtr(int p,int n)
{
    int pval;
    for (pval = 1; pval < n; pval++)
    {
        st[p] = pval;
        if (valid(p))
        {
            if (p == n) tipar(p);
            else backtr(p + 1,n);
        }
    }
}

public void tipar(int p)
{
    for (int i = 1; i < p; i++)
    {
        f3 += st[i] + " ";
    }
    f3 += "\n";
}

```

```

public static bool valid(int p)
{
    int i;
    bool ok=true;
    for (i = 1; i < p - 1; i++)
        if (st[p] == st[i]) ok = false;
    return ok;
}

```

```

public static void afisare(int[,] a, int n)
{
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            Console.Write(a[i, j]);
        }
        Console.WriteLine();
    }
    Console.WriteLine();
}

```

```

public static void asociativ(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[j, k];
                d1 = masiv[i, j];
                if (masiv[i, d] != masiv[d1, k]) l++;
            }
        }
    }
    if (l == 0) Console.WriteLine("ESTE ASOCIATIV");
    else Console.WriteLine("NU ESTE ASOCIATIV");
}

```

```

public static void medial(int[,] masiv, int n)
{
    int l = 0;

    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                for (int t = 1; t < n + 1; t++)
                {

```

```

        d = masiv[i, j];
        r = masiv[k, t];
        d1 = masiv[i, k];
        r1 = masiv[j, t];
        if (masiv[d, r] != masiv[d1, r1])
            l++;
    }
}

}

if (l == 0) Console.WriteLine("ESTE MEDIAL");
else Console.WriteLine("NU ESTE MEDIAL");

}

public static void paramedial(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                for (int t = 1; t < n + 1; t++)
                {

                    d = masiv[i, j];
                    r = masiv[k, t];
                    d1 = masiv[t, j];
                    r1 = masiv[k, i];
                    if (masiv[d, r] != masiv[d1, r1]) l++;

                }
            }
        }
    }
    if (l == 0) Console.WriteLine("ESTE PARAMEDIAL");
    else Console.WriteLine("NU ESTE PARAMEDIAL");
}

public static void bicomutativ(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                for (int t = 1; t < n + 1; t++)
                {

                    d = masiv[i, j];
                    r = masiv[k, t];
                    d1 = masiv[t, k];
                    r1 = masiv[j, i];
                    if (masiv[d, r] != masiv[d1, r1]) l++;

                }
            }
        }
    }
}

```

```

    }
}

if (l == 0) Console.WriteLine("ESTE BICOMUTATIV");
else Console.WriteLine("NU ESTE BICOMUTATIV");
}

public static void ag_gr(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[k, j];
                if (masiv[d, k] != masiv[d1, i]) l++;
            }
        }
    }
    if (l == 0) Console.WriteLine("ESTE AG GRUPOID");
    else Console.WriteLine("NU ESTE AG GRUPOID");
}

public static void ga_gr(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[j, i];
                if (masiv[d, k] != masiv[k, d1]) l++;
            }
        }
    }
    if (l == 0) Console.WriteLine("ESTE GA GRUPOID");
    else Console.WriteLine("NU ESTE GA GRUPOID");
}

public static void ga_gr1(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[k, i];
                if (masiv[d, k] != masiv[d1, j]) l++;
            }
        }
    }
}

```

```

    }
    if (l == 0) Console.WriteLine("ESTE GA1 GRUPOID");
    else Console.WriteLine("NU ESTE GA1 GRUpoid");
}

```

```

public static void ad_gr(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[j, k];
                d1 = masiv[j, i];
                if (masiv[i, d] != masiv[k, d1]) l++;
            }
        }
    }
    if (l == 0) Console.WriteLine("ESTE AD GRUPOID");
    else Console.WriteLine("NU ESTE AD GRUpoid");
}

```

```

public static void da_gr(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[j, k];
                d1 = masiv[i, j];
                if (masiv[i, d] != masiv[k, d1]) l++;
            }
        }
    }
    if (l == 0) Console.WriteLine("ESTE DA GRUPOID");
    else Console.WriteLine("NU ESTE DA GRUpoid");
}

```

```

public static void hexagonal(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
    {
        for (int j = 1; j < n + 1; j++)
        {
            for (int k = 1; k < n + 1; k++)
            {
                for (int t = 1; t < n + 1; t++)
                {
                    d = masiv[i, j];
                    r = masiv[k, t];
                    d1 = masiv[i, k];
                    r1 = masiv[j, t];

```



```

        r2 = masiv[j, i];
        if (masiv[i, i] != i || masiv[d, r] != masiv[d1, r1] ||
masiv[i, r2] != masiv[d, i] && masiv[d, i] != j) l++;
    }
}
}
if (l == 0) Console.WriteLine("ESTE HEXAGONAL");
else Console.WriteLine("NU ESTE HEXAGONAL");
}

```

```

public static void dist_dr(int[,] masiv, int n)
{
    int l = 0;

    for (int i = 1; i < n + 1; i++)
        for (int j = 1; j < n + 1; j++)
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[i, k];
                r1 = masiv[j, k];
                if (masiv[d, k] != masiv[d1, r1]) l++;
            }

    if (l == 0) Console.WriteLine("ESTE DISTRIBUTIV DE DREAPTA");
    else Console.WriteLine("NU ESTE DISTRIBUTIV DE DREAPTA");
}

```

```

public static void dist_st(int[,] masiv, int n)
{
    int l = 0;

    for (int i = 1; i < n + 1; i++)
        for (int j = 1; j < n + 1; j++)
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[k, i];
                r1 = masiv[k, j];
                if (masiv[k, d] != masiv[d1, r1]) l++;
            }

    if (l == 0) Console.WriteLine("ESTE DISTRIBUTIV DE STANGA");
    else Console.WriteLine("NU ESTE DISTRIBUTIV DE STANGA");
}

```

```

public static void unitate_dreapta(int[,] masiv, int n, out int r)
{

```

```

int l;
int j = 0; r = 0;
for (int i = 1; i < n + 1; i++)
{
    l = 0;
    j++;
    if (masiv[j, i] == i)
    {
        for (int k = 1; k < n + 1; k++)
        {
            if (masiv[k, j] == k) l++;
            if (l == n) r = j;
        }
    }
}
if (r != 0) Console.WriteLine("ESTE UNITATE DREAPTA " + r);
else Console.WriteLine("NU ESTE UNITATE STANGA");
}

```

```

public static void unitate_stanga(int[,] masiv, int n, out int r2)
{
    int l;
    int j = 0; r2 = 0;
    for (int i = 1; i < n + 1; i++)
    {
        l = 0;
        j++;
        if (masiv[i, j] == i)
        {
            for (int k = 1; k < n + 1; k++)
            {
                if (masiv[j, k] == k) l++;
                if (l == n) r2 = j;
            }
        }
    }
    if (r2 != 0) Console.WriteLine("ESTE UNITATE STANGA " + r2);
    else Console.WriteLine("NU ESTE UNITATE STANGA");
}

```

```

public static void unitate(ref int r, ref int r2)
{
    if (r1 == r2 && r > 0) Console.WriteLine("Este unitate " + r);
    else Console.WriteLine("NU este unitate");
}

```

```

public static void ward(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
        for (int j = 1; j < n + 1; j++)
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[i, k];
                d2 = masiv[j, k];
                if (d != masiv[d1, d2]) l += 1;
            }
}

```

```

    }

    if (l == 0) Console.WriteLine("ESTE WARD");
    else Console.WriteLine("NU ESTE WARD");
}

public static void ward_invers(int[,] masiv, int n)
{
    int l = 0;
    for (int i = 1; i < n + 1; i++)
        for (int j = 1; j < n + 1; j++)
            for (int k = 1; k < n + 1; k++)
            {
                d = masiv[i, j];
                d1 = masiv[k, i];
                d2 = masiv[k, j];
                if (d != masiv[d1, d2]) l += 1;
            }

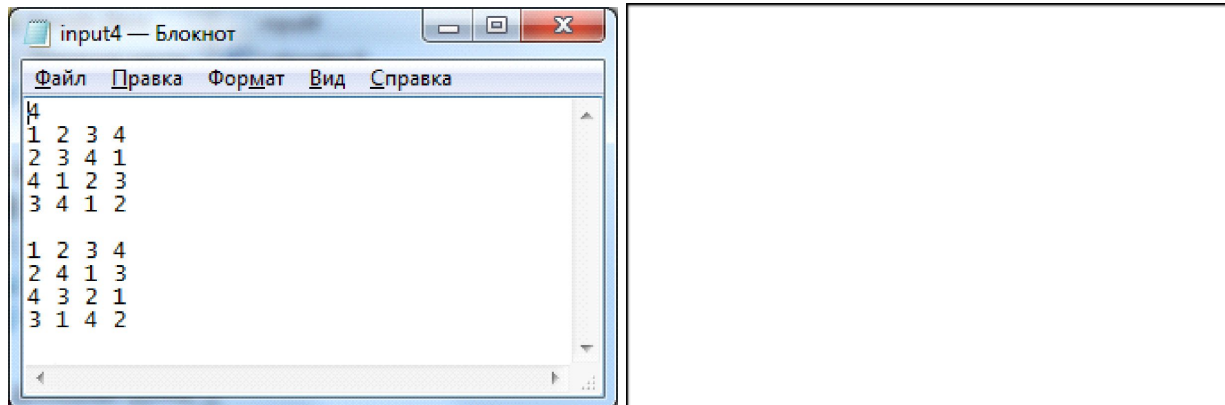
    if (l == 0) Console.WriteLine("ESTE WARD INVERS");
    else Console.WriteLine("NU ESTE WARD INVERS");
}
}
}

```

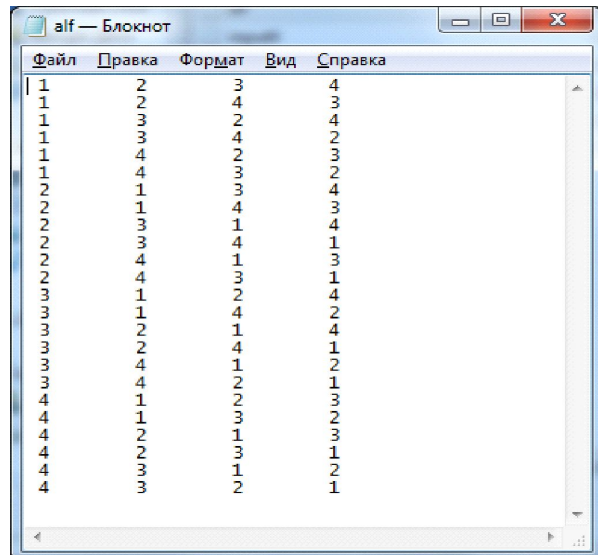
În urma compilării am obținem următoare rezultate :

Ex.1

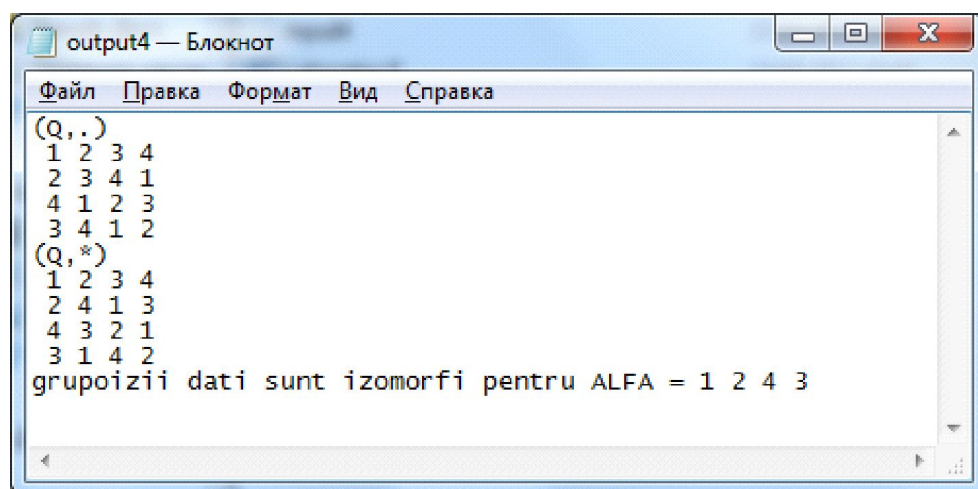
Fișierul input4.txt



Fișierul alf.txt

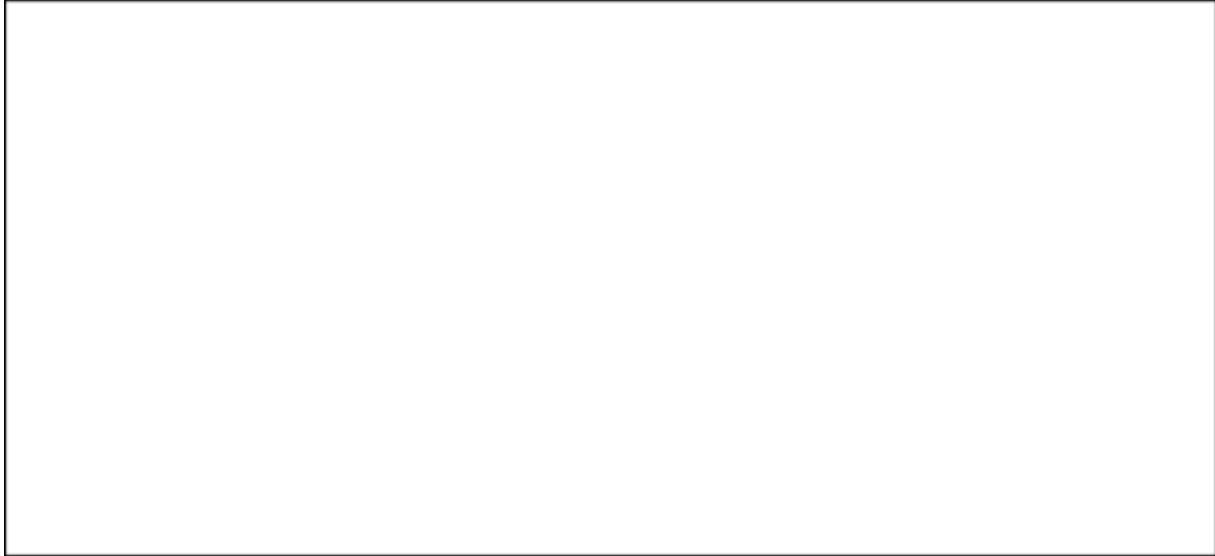


Fișierul output4.txt



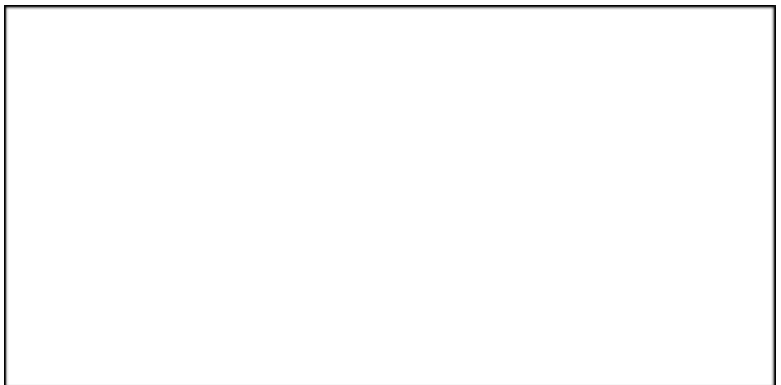
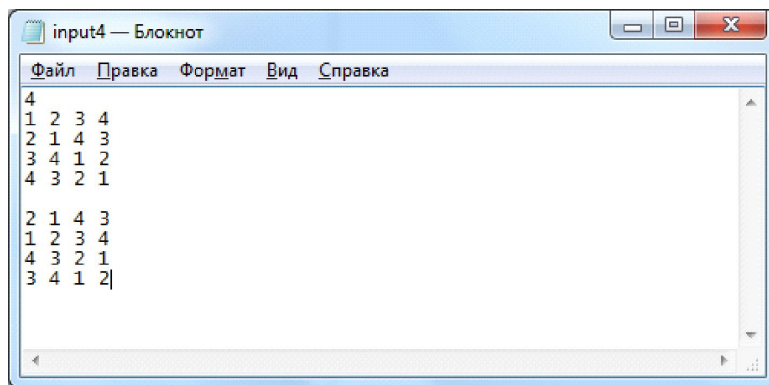
Proprietăți:

Pentru quasigrupul	avem proprietatile	Pentru quasigrupul izotop	avem proprietatile
Nu este asociativ		Nu este asociativ	
Nu este medial		Nu este medial	
Nu este paramedial		Nu este paramedial	
Nu este bicomutativ		Nu este bicomutativ	
Nu este AG grupoid		Nu este AG grupoid	
Nu este GA grupoid		Nu este GA grupoid	
Nu este GA1 grupoid		Nu este GA1 grupoid	
Nu este AD grupoid		Nu este AD grupoid	
Nu este DA grupoid		Nu este DA grupoid	
Nu este hexagonal		Nu este hexagonal	
Nu este distributiv la dreapta		Nu este distributiv la dreapta	
Nu este distributiv la stanga		Nu este distributiv la stanga	
Nu este unitate de dreapta		Nu este unitate de dreapta	
Este unitate de stanga 1		Este unitate de stanga 1	
Nu este unitate		Nu este unitate	
Nu este Ward		Nu este Ward	
Nu este Ward invers		Nu este Ward invers	



Ex.2

Fişierul input4.txt



Fişierul alf.txt

alf — Блокнот

Файл	Правка	Формат	Вид	Справка
1	2	3	4	
1	2	4	3	
1	3	2	4	
1	4	3	2	
1	4	3	2	
2	1	3	4	
2	1	4	3	
2	3	1	4	
2	3	1	4	
2	4	1	3	
2	4	3	1	
3	1	2	4	
3	2	1	4	
3	4	1	2	
3	4	2	1	
4	1	2	3	
4	1	3	2	
4	2	1	3	
4	3	1	2	
4	3	2	1	

Fișierul output4.txt

output4 — Блокнот

```

Файл  Правка  Формат  Вид  Справка
| (Q, .)
1 2 3 4
2 1 4 3
3 4 1 2
4 3 2 1
| (Q, *)
2 1 4 3
1 2 3 4
4 3 2 1
3 4 1 2
grupoizii dati sunt izomorfi pentru ALFA = 2 1 3 4
grupoizii dati sunt izomorfi pentru ALFA = 2 1 4 3
grupoizii dati sunt izomorfi pentru ALFA = 2 3 4 1
grupoizii dati sunt izomorfi pentru ALFA = 2 4 1 3

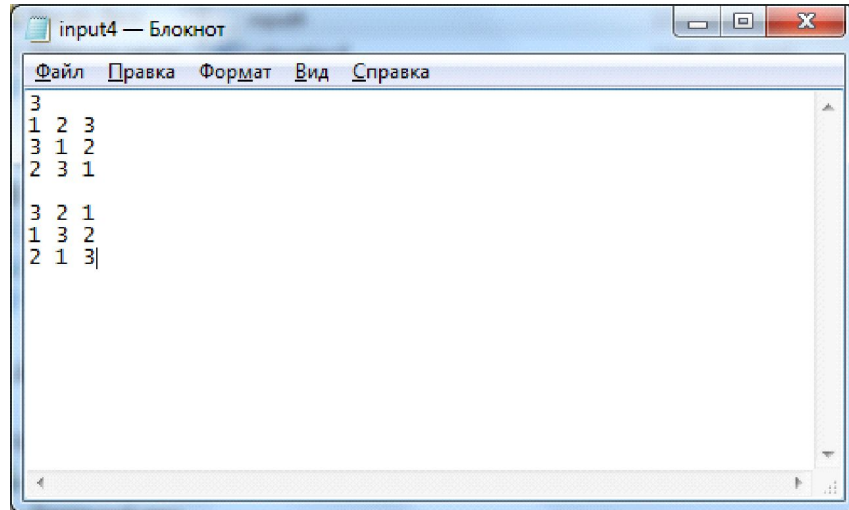
```

Proprietăți

Pentru quasigrupul avem proprietatile	Pentru quasigrupul izotop avem proprietatile
Este asociativ	Este asociativ
Este medial	Este medial
Este paramedial	Este paramedial
Este bicomutativ	Este bicomutativ
Este AG grupoid	Este AG grupoid
Este GA grupoid	Este GA grupoid
Este GA1 grupoid	Este GA1 grupoid
Este AD grupoid	Este AD grupoid
Este DA grupoid	Este DA grupoid
Nu este hexagonal	Nu este hexagonal
Nu este distributiv la dreapta	Nu este distributiv la dreapta
Nu este distributiv la stanga	Nu este distributiv la stanga
Este unitate de dreapta 1	Este unitate de dreapta 2
Este unitate de stanga 1	Este unitate de stanga 2
Este unitate 1	Este unitate 2
Este Ward	Este Ward
Este Ward invers	Este Ward invers

Ex.3

Fișierul input4.txt

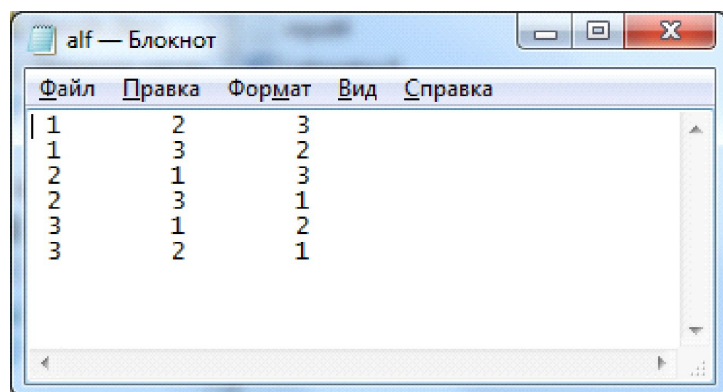


```
3
1 2 3
3 1 2
2 3 1

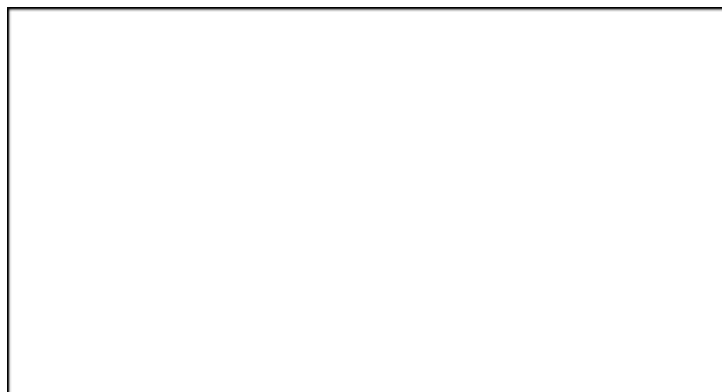
3 2 1
1 3 2
2 1 3|
```



Fișierul alf.txt



```
| 1      2      3
1      3      2
2      1      3
2      3      1
3      1      2
3      2      1
```



Fișierul output4.txt

The screenshot shows a Notepad window with the following content:

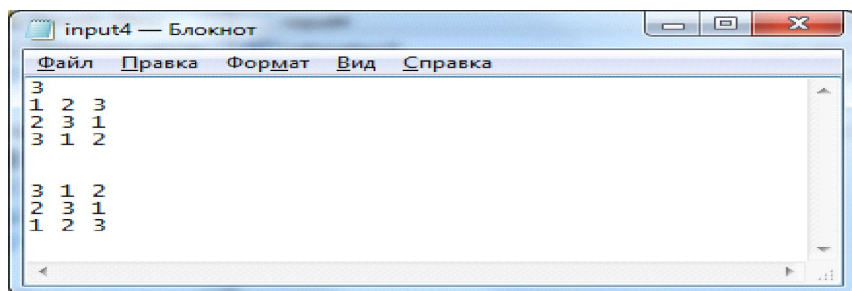
```
Файл  Правка  Формат  Вид  Справка
|(Q,.)
1 2 3
3 1 2
2 3 1
(Q,*)
3 2 1
1 3 2
2 1 3
grupoizii dati nu sunt izomorfi
```

Proprietăți

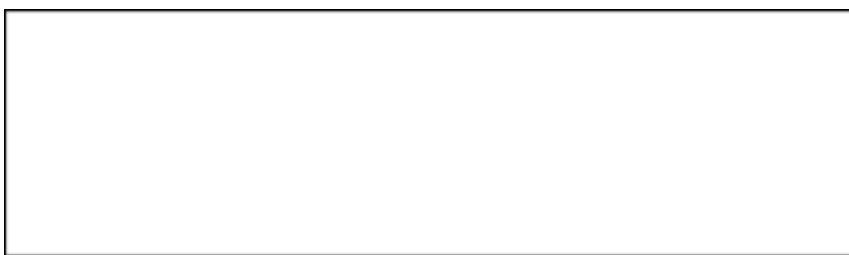
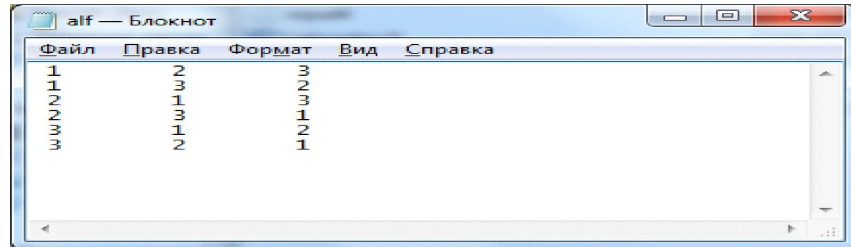
Pentru quasigrupul avem proprietatile	Pentru quasigrupul izotop avem proprietatile
Nu este asociativ	Nu este asociativ
Este medial	Este medial
Este paramedial	Este paramedial
Este bicomutativ	Este bicomutativ
Este AG grupoid	Nu este AG grupoid
Nu este GA grupoid	Nu este GA grupoid
Nu este GA1 grupoid	Nu este GA1 grupoid
Nu este AD grupoid	Este AD grupoid
Nu este DA grupoid	Nu este DA grupoid
Nu este hexagonal	Nu este hexagonal
Nu este distributiv la dreapta	Nu este distributiv la dreapta
Nu este distributiv la stanga	Nu este distributiv la stanga
Nu este unitate de dreapta	Este unitate de dreapta 3
Este unitate de stanga 1	Nu este unitate de stanga
Nu este unitate	Nu este unitate
Nu este Ward	Este Ward
Este Ward invers	Nu este Ward invers

Ex.4

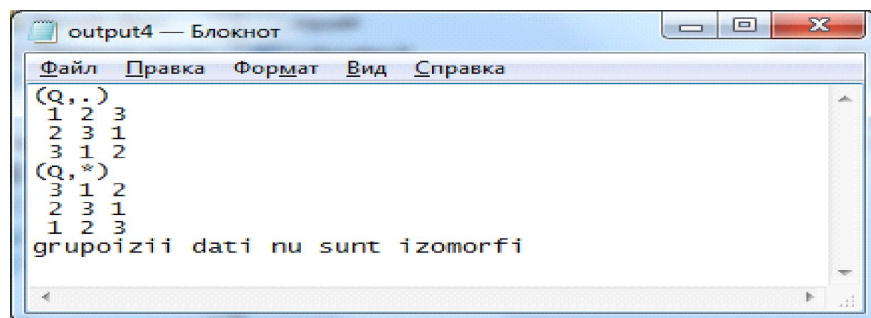
Fișierul input4.txt



Fișierul alf.txt



Fișierul output4.txt



Proprietăți:

Pentru quasigrupul avem proprietatile Pentru quasigrupul izotop avem proprietatile

Este asociativ	Nu este asociativ
Este medial	Este medial
Este paramedial	Este paramedial
Este bicomutativ	Este bicomutativ
Este AG grupoid	Este AG grupoid
Este GA grupoid	Nu este GA grupoid
Este GA1 grupoid	Nu este GA1 grupoid
Este AD grupoid	Nu este AD grupoid
Este DA grupoid	Nu este DA grupoid
Nu este hexagonal	Nu este hexagonal
Nu este distributiv la dreapta	Nu este distributiv la dreapta
Nu este distributiv la stanga	Nu este distributiv la stanga
Este unitate de dreapta 1	Nu este unitate de dreapta
Este unitate de stanga 1	Este unitate de stanga 3
Este unitate 1	Nu este unitate
Nu este Ward	Nu este Ward
Nu este Ward invers	Este Ward invers

Tabelul rezultatelor obținute :

	Ex.1		Ex.2		Ex.3		Ex.4	
Avem izomorfism pentru substituția	$\alpha = 1\ 2\ 4\ 3$		$\alpha = 2\ 1\ 3\ 4$ $\alpha = 2\ 1\ 4\ 3$ $\alpha = 2\ 3\ 4\ 1$ $\alpha = 2\ 4\ 1\ 3$		Nu		Nu	
Asociativ	-	-	+	+	-	-	+	-
Medial	-	-	+	+	+	+	+	+
Paramedial	-	-	+	+	+	+	+	+
Bicomutativ	-	-	+	+	+	+	+	+
AG grupoid	-	-	+	+	+	-	+	+
GA grupoid	-	-	+	+	-	-	+	-
GA1 grupoid	-	-	+	+	-	-	+	-
AD grupoid	-	-	+	+	-	+	+	-
DA grupoid	-	-	+	+	-	-	+	-
Hexagonal	-	-	-	-	-	-	-	-
Distributiv la dreapta	-	-	-	-	-	-	-	-
Distributiv la stanga	-	-	-	-	-	-	-	-
Unitate de dreapta	-	-	1	2	-	3	1	-
Unitate de stanga	1	1	1	2	1	-	1	3
Unitate	-	-	1	2	-	-	1	-
Ward	-	-	+	+	-	+	-	-
Ward invers	-	-	+	+	+	-	-	+

