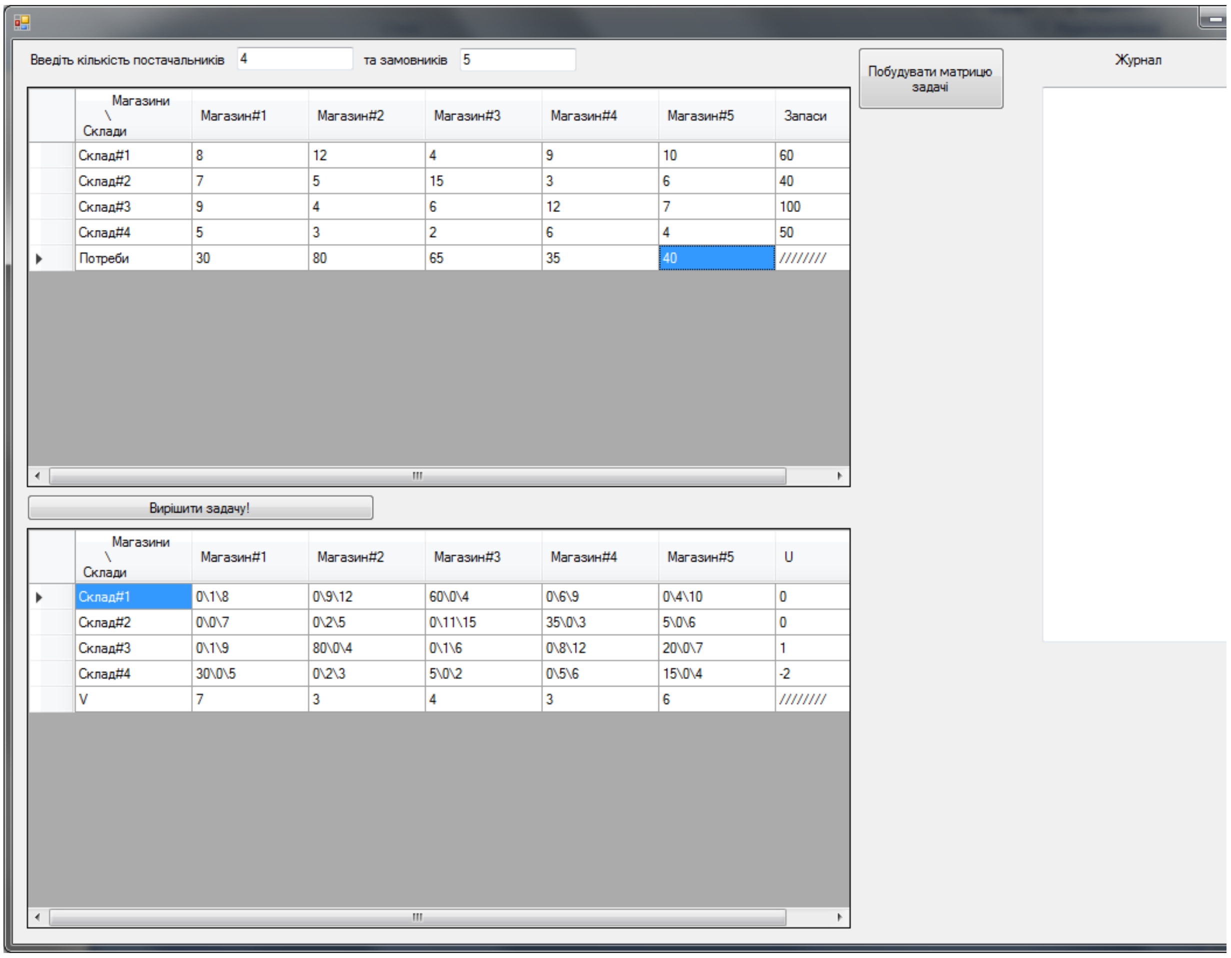
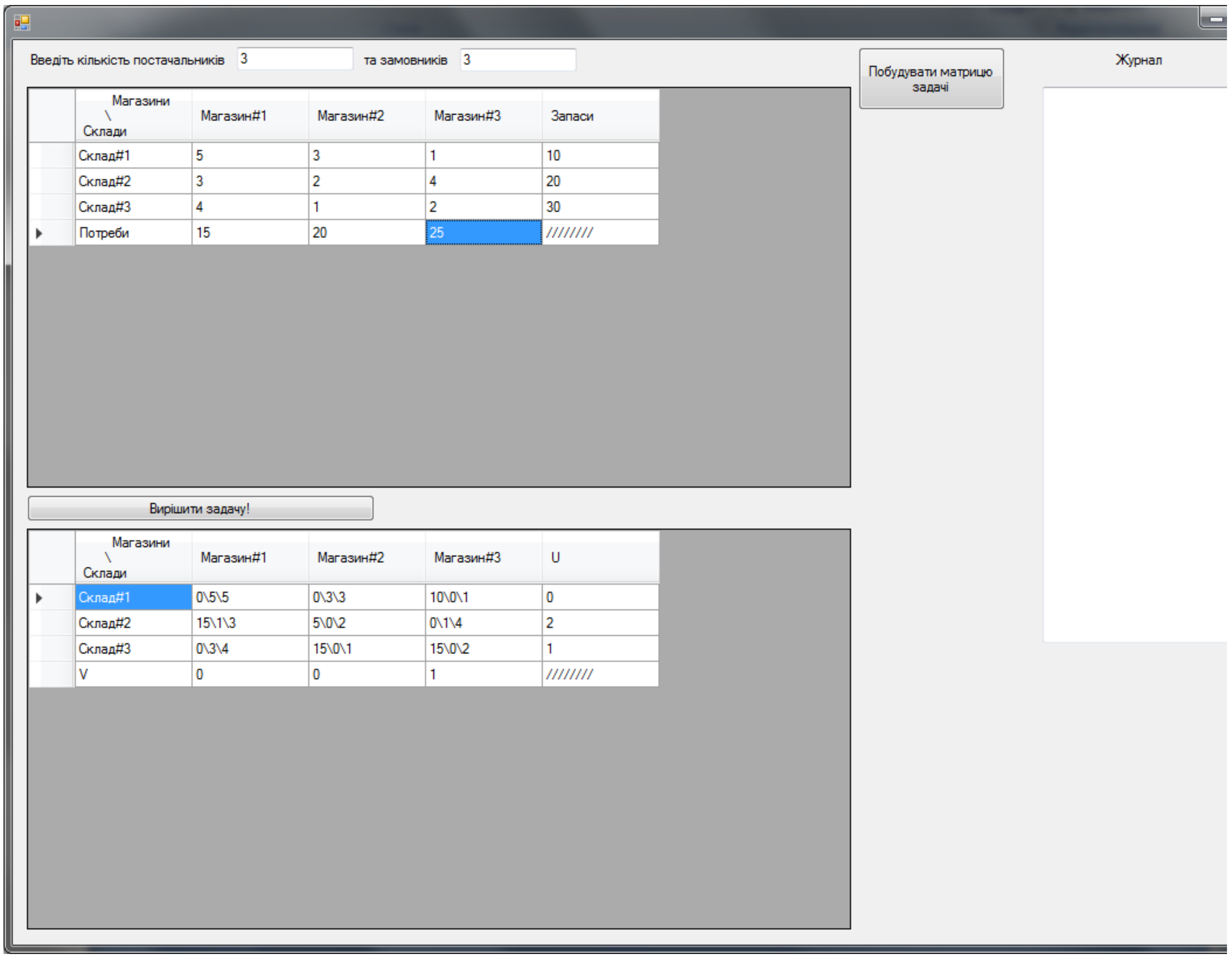


Введіть кількість постачальників 4 та замовників 5

Магазини \ Склади	Магазин#1	Магазин#2	Магазин#3	Магазин#4	Магазин#5	Запаси
Склад#1	8	12	4	9	10	60
Склад#2	7	5	15	3	6	40
Склад#3	9	4	6	12	7	100
Склад#4	5	3	2	6	4	50
Потреби	30	80	65	35	40	////////

Магазини \ Склади	Магазин#1	Магазин#2	Магазин#3	Магазин#4	Магазин#5	U
Склад#1	0\1\8	0\9\12	60\0\4	0\6\9	0\4\10	0
Склад#2	0\0\7	0\2\5	0\11\15	35\0\3	5\0\6	0
Склад#3	0\1\9	80\0\4	0\1\6	0\8\12	20\0\7	1
Склад#4	30\0\5	0\2\3	5\0\2	0\5\6	15\0\4	-2
V	7	3	4	3	6	////////



Введіть кількість постачальників 3 та замовників 3

Магазини \ Склади	Магазин#1	Магазин#2	Магазин#3	Запаси
Склад#1	5	3	1	10
Склад#2	3	2	4	20
Склад#3	4	1	2	30
Потреби	15	20	25	////////

Магазини \ Склади	Магазин#1	Магазин#2	Магазин#3	U
Склад#1	0\5\5	0\3\3	10\0\1	0
Склад#2	15\1\3	5\0\2	0\1\4	2
Склад#3	0\3\4	15\0\1	15\0\2	1
V	0	0	1	////////

```
int providers, purchasers;
            int i, j, k;
            providers = dataGridView1.Rows.Count - 1;
            purchasers = dataGridView1.Columns.Count - 2;
            if (dataGridView1.Columns.Count < 1 ||
dataGridView1.Rows[providers].Cells[purchasers + 1].Value != "////////")
            {
                textBox3.Text += "Task is not complete....yet";
                return;
            }
            element[] matrix = new element[providers* purchasers];
            double[] Holdings=new double[providers],  Needs  = new double[purchasers];
            double HoldingsSum = 0, NeedsSum = 0;
            try
            {
                for (i = 0; i < providers; i++)
                {
                    for (j = 1; j <= purchasers;j++)
                    {
                        matrix[i*purchasers+ j - 1].Cost =
Convert.ToDouble(dataGridView1.Rows[i].Cells[j].Value);
                        if (matrix[i * purchasers + j - 1].Cost <= 0)
                        {
                            throw new Exception();
                        }
                        matrix[i * purchasers + j - 1].x = j - 1;
                        matrix[i * purchasers + j - 1].y = i;
                    }
                    Holdings[i] =
Convert.ToDouble(dataGridView1.Rows[i].Cells[purchasers+1].Value);
                    if (Holdings[i] < 0)
                    {
                        throw new Exception();
                    }
                    HoldingsSum += Holdings[i];
                }
                for (j = 1; j <= purchasers; j++)
                {
                    Needs[j - 1] =
Convert.ToDouble(dataGridView1.Rows[providers].Cells[j].Value);
                    if (Needs[j - 1] < 0)
                    {
                        throw new Exception();
                    }
                    NeedsSum+=Needs[j-1];
                }
            }
            catch
            {
                textBox3.Text += "Wrong input, mortal"+Environment.NewLine;
                return;
            }
            if(Math.Round(HoldingsSum,10)!=Math.Round(NeedsSum,10))
            {
                textBox3.Text += "Problem shall be closed" + Environment.NewLine;
                return;
            }
            matrix = matrix.OrderBy(Temp => Temp.Cost).ToArray();
            element[,] RelienceMatrix = new element[providers, purchasers];
            double[] ResultNeeds = new double[purchasers];
            double[] ResultHoldings = new double[providers];
            for (i = 0; i < providers; i++)
            {
                ResultHoldings[i] = Holdings[i];
                for (j = 0; j < providers; j++)
                {
                    RelienceMatrix[i, j]=new element(i,j);
                }
            }
            for (i = 0; i < purchasers; i++)
            {
                ResultNeeds[i] = Needs[i];
            }
            int BasicCells = 0;
            for (i = 0; i < matrix.Length; i++)
            {
                RelienceMatrix[matrix[i].y, matrix[i].x].Cost = matrix[i].Cost;
                if (ResultNeeds[matrix[i].x] < ResultHoldings[matrix[i].y])
                {
                    RelienceMatrix[matrix[i].y, matrix[i].x].Value = 
ResultNeeds[matrix[i].x];
                    ResultHoldings[matrix[i].y] -= ResultNeeds[matrix[i].x];
                    ResultNeeds[matrix[i].x] = 0;
                    if (RelienceMatrix[matrix[i].y, matrix[i].x].Value != 0)
                    {
                        BasicCells++;
                    }
                }
                else
                {
                    RelienceMatrix[matrix[i].y, matrix[i].x].Value = 
ResultHoldings[matrix[i].y];
                    ResultNeeds[matrix[i].x] -= ResultHoldings[matrix[i].y];
                    ResultHoldings[matrix[i].y] = 0;
                    if (RelienceMatrix[matrix[i].y, matrix[i].x].Value != 0)
                    {
                        BasicCells++;
                    }
                }
            }
            if (BasicCells < providers + purchasers - 1)
            {
                textBox3.Text += "Degeneracy relience matrix" + Environment.NewLine;
                return;
            }
            double[] ProvPotent = new double[providers], PurchPotent = new 
double[purchasers];
            bool[] ProvPotentNum = new bool[providers], PurchPotentNum = new 
bool[purchasers];
            int AllPotentUnupdatedNum = providers + purchasers - 1;
            for (j = 0; j < providers; j++)
            {
                ProvPotentNum[j] = false;
            }
            for (i = 0; i < purchasers; i++)
            {
                PurchPotentNum[i] = false;
            }
            ProvPotentNum[0] = true;
            ProvPotent[0] = 0;
            AllPotentUnupdatedNum--;
            while (AllPotentUnupdatedNum > 0)
            {
                for (j = 0; j < providers; j++)
                {
                    if (ProvPotentNum[j] == true)
                    {
                        for (i = 0; i < purchasers; i++)
                        {
                            if ((RelienceMatrix[j,i].Value != 0)&&(PurchPotentNum[i]
==false))
                            {
                                PurchPotent[i] = RelienceMatrix[j, i].Cost -
ProvPotent[j];
                                PurchPotentNum[i] = true;
                                AllPotentUnupdatedNum--;
                            }
                        }
                    }
                }
                for (j = 0; j < purchasers; j++)
                {
                    if (PurchPotentNum[j] == true)
                    {
                        for (i = 0; i < providers; i++)
                        {
                            if ((RelienceMatrix[i, j].Value != 0)&&(ProvPotentNum[i]
==false))
                            {
                                ProvPotent[i] = RelienceMatrix[i, j].Cost -
PurchPotent[j];
                                ProvPotentNum[i] = true;
                                AllPotentUnupdatedNum--;
                            }
                        }
                    }
                }
            }
            for (i = 0; i < providers; i++)
            {
                for (j = 0; j < purchasers; j++)
                {
                    RelienceMatrix[i, j].delta = RelienceMatrix[i, j].Cost -
ProvPotent[i] - PurchPotent[j];
                }
            }
            bool TimeToEnd = true;
            int wrongx = -1, wrongy = -1;
            for (i = 0; i < providers; i++)
            {
                for (j = 0; j < purchasers; j++)
                {
                    if (RelienceMatrix[i, j].delta < 0)
                    {
                        TimeToEnd = false;
                        wrongx = j;
                        wrongy = i;
                    }
                }
            }
            int Tempx = -1, Tempy = -1;
            double temp;
            int NumberOfIter = 1;
            while (!TimeToEnd)
            {
                for (i = 0; i < providers; i++)
                {
                    if (RelienceMatrix[i, wrongx].Value > 0)
                    {
                        for (j = 0; j < purchasers; j++)
                        {
                            if (RelienceMatrix[wrongy, j].Value > 0)
                            {
                                if (RelienceMatrix[i, j].Value > 0)
                                {
                                    Tempx = j;
                                    Tempy = i;
                                    break;
                                }
                            }
                        }
                        if (Tempx!=-1)
                            break;
                    }
                }
                temp = Math.Min(RelienceMatrix[Tempy, wrongx].Value,
RelienceMatrix[wrongy, Tempx].Value);
                RelienceMatrix[Tempy, wrongx].Value -= temp;
                RelienceMatrix[wrongy, Tempx].Value-= temp;
                RelienceMatrix[Tempy, Tempx].Value+= temp;
                RelienceMatrix[wrongy, wrongx].Value += temp;
                AllPotentUnupdatedNum = providers + purchasers - 1;
                for (j = 0; j < providers; j++)
                {
                    ProvPotentNum[j] = false;
                }
                for (i = 0; i < purchasers; i++)
                {
                    PurchPotentNum[i] = false;
                }
                ProvPotentNum[0] = true;
                ProvPotent[0] = 0;
                AllPotentUnupdatedNum--;
                while (AllPotentUnupdatedNum > 0)
                {
                    for (j = 0; j < providers; j++)
                    {
                        if (ProvPotentNum[j] == true)
                        {
                            for (i = 0; i < purchasers; i++)
                            {
                                if ((RelienceMatrix[j, i].Value != 0) &&
(PurchPotentNum[i] == false))
                                {
                                    PurchPotent[i] = RelienceMatrix[j, i].Cost -
ProvPotent[j];

                                    PurchPotentNum[i] = true;
                                    AllPotentUnupdatedNum--;
                                }
                            }
                        }
                    }
                    for (j = 0; j < purchasers; j++)
                    {
                        if (PurchPotentNum[j] == true)
                        {
                            for (i = 0; i < providers; i++)
                            {
                                if ((RelienceMatrix[i, j].Value != 0) &&
(ProvPotentNum[i] == false))
                                {
                                    ProvPotent[i] = RelienceMatrix[i, j].Cost -
PurchPotent[j];

                                    ProvPotentNum[i] = true;
                                    AllPotentUnupdatedNum--;
                                }
                            }
                        }
                    }
                }
                for (i = 0; i < providers; i++)
                {
                    for (j = 0; j < purchasers; j++)
                    {
                        RelienceMatrix[i, j].delta = RelienceMatrix[i, j].Cost -
ProvPotent[i] - PurchPotent[j];
                    }
                }
                TimeToEnd = true;
                wrongx = -1; wrongy = -1;
                for (i = 0; i < providers; i++)
                {
                    for (j = 0; j < purchasers; j++)
                    {
                        if (RelienceMatrix[i, j].delta < 0)
                        {
                            TimeToEnd = false;
                            wrongx = j;
                            wrongy = i;
                        }
                    }
                }
                Tempx = -1; Tempy = -1;
                NumberOfIter++;
            }
            Console.WriteLine(NumberOfIter);
            dataGridView2.Rows.Clear();
            dataGridView2.Columns.Clear();
            dataGridView2.Columns.Add(new DataGridViewTextBoxColumn());
            dataGridView2.Columns[0].ReadOnly = false;
            dataGridView2.Columns[0].SortMode = DataGridViewColumnSortMode.NotSortable;
            dataGridView2.Columns[0].HeaderText = ("        Магазини\n      \\
\nСклади");
            dataGridView2.Columns[0].Name = "Column0";
            for (i = 1; i <= purchasers; i++)
            {
                dataGridView2.Columns.Add(new DataGridViewTextBoxColumn());
                dataGridView2.Columns[i].HeaderText = ("Магазин#" + i);
                dataGridView2.Columns[i].ReadOnly = false;
                dataGridView2.Columns[i].SortMode =
DataGridViewColumnSortMode.NotSortable;
                dataGridView2.Columns[0].Name = "Column" + i;
            }
            dataGridView2.Columns.Add(new DataGridViewTextBoxColumn());
            dataGridView2.Columns[purchasers + 1].HeaderText = "U";
            dataGridView2.Columns[purchasers + 1].ReadOnly = false;
            dataGridView2.Columns[purchasers + 1].SortMode =
DataGridViewColumnSortMode.NotSortable;
            dataGridView2.Columns[0].Name = "Column" + i;
            for (i = 0; i < providers; i++)
            {
                dataGridView2.Rows.Add(new DataGridViewRow());
                dataGridView2.Rows[i].Cells[0].Value = "Склад#" + (i + 1);
            }
            dataGridView2.Rows.Add(new DataGridViewColumn());
            dataGridView2.Rows[providers].Cells[0].Value = "V";// +(i + 1);
            dataGridView2.Rows[providers].Cells[purchasers + 1].Value = "///////";
            dataGridView2.Rows[providers].Cells[purchasers + 1].ReadOnly = false;
            for (i = 0; i < providers; i++)
            {
                for (j = 0; j < purchasers; j++)
                {
                    dataGridView2.Rows[i].Cells[j + 1].Value = "" + RelienceMatrix[i,
j].Value + "\\" + RelienceMatrix[i, j].delta + "\\" + RelienceMatrix[i,j].Cost;
                }
                dataGridView2.Rows[i].Cells[purchasers + 1].Value = ProvPotent[i];
            }
            for (j = 0; j < purchasers; j++)
            {
                dataGridView2.Rows[providers].Cells[j+1].Value = PurchPotent[j];
            }
```