# **North West Corner Method – (C++ Language)**

### **Code**

#include <iostream>

#include <stdio.h>

#include <conio.h>

#include <iomanip>

#include <stdlib.h>

#define MAX 50

using namespace std;

enum boolean

{

    FALSE,

    TRUE

};

class nwcmethod

{

    int data[MAX][MAX];

    int requered[MAX];

    int capacity[MAX];

    int allocation[MAX][MAX];

    int no\_of\_rows, no\_of\_columns, no\_of\_allocation;

    public:

        nwcmethod()

        {

            for (int i = 0; i < MAX; i++)

            {

                capacity[i] = 0;

                requered[i] = 0;

                for (int j = 0; j < MAX; j++)

                {

                    data[i][j] = 0;

                    allocation[i][j] = 0;

                }

            }

            no\_of\_rows = no\_of\_columns = no\_of\_allocation = 0;

        }

    void setColumn(int no)

    {

        no\_of\_columns = no;

    };

    void setRow(int no)

    {

        no\_of\_rows = no;

    }

    void getData();

    void getCapacity();

    void getRequiredValue();

    void makeAllocation();

    boolean checkValue(int[], int);

    void display();

};

boolean nwcmethod::checkValue(int arr[], int no)

{

    for (int i = 0; i < no; i++)

        if (arr[i] != 0)

            return FALSE;

    return TRUE;

}

void arrayCopy(int start, int end, int array1[], int start1, int array2[])

{

    for (int i = start, j = start1; i < end; i++, j++)

    {

        array2[j] = array1[i];

    }

}

int getTotal(int array[], int no)

{

    int sum = 0;

    for (int i = 0; i < no; i++)

        sum += array[i];

    return sum;

}

void nwcmethod::makeAllocation()

{

    int i = 0, j = 0;

    int temp\_requered[MAX] = { 0 };

    int temp\_capacity[MAX] = { 0 };

    int sum\_of\_cap, sum\_of\_req;

    sum\_of\_cap = getTotal(capacity, no\_of\_rows);

    sum\_of\_req = getTotal(requered, no\_of\_columns);

    if (sum\_of\_cap != sum\_of\_req)

    {

        if (sum\_of\_cap > sum\_of\_req)

        {

            for (j = 0; j < no\_of\_rows; j++)

                data[j][no\_of\_columns] = 0;

            requered[no\_of\_columns] = sum\_of\_cap - sum\_of\_req;

            no\_of\_columns++;

        }

        else

        {

            for (j = 0; j < no\_of\_columns; j++)

                data[no\_of\_rows][j] = 0;

            capacity[no\_of\_rows] = sum\_of\_req - sum\_of\_cap;

            no\_of\_rows++;

        }

    }

    i = j = 0;

    arrayCopy(0, no\_of\_rows, capacity, 0, temp\_capacity);

    arrayCopy(0, no\_of\_columns, requered, 0, temp\_requered);

    while (!checkValue(temp\_capacity, no\_of\_rows) || !checkValue(temp\_requered, no\_of\_columns))

    {

        if (temp\_capacity[i] > temp\_requered[j])

        {

            allocation[i][j] = temp\_requered[j];

            temp\_capacity[i] -= temp\_requered[j];

            temp\_requered[j] = 0;

            j++;

        }

        else if (temp\_capacity[i] < temp\_requered[j])

        {

            allocation[i][j] = temp\_capacity[i];

            temp\_requered[j] -= temp\_capacity[i];

            temp\_capacity[i] = 0;

            i++;

        }

        else

        {

            allocation[i][j] = temp\_capacity[i];

            temp\_capacity[i] = temp\_requered[j] = 0;

            i++;

            j++;

        }

        no\_of\_allocation++;

    }

}

void nwcmethod::getCapacity()

{

    cout << "==================== Enter Supply ====================\n\n";

    for (int i = 0; i < no\_of\_rows; i++)

    {

        cout << "Supply\_#" << i + 1 << ": ";

        cin >> capacity[i];

    }

    cout << endl;

}

void nwcmethod::getRequiredValue()

{

    cout << "==================== Enter Demand ====================\n\n";

    for (int i = 0; i < no\_of\_columns; i++)

    {

        cout << "Demand\_#" << i + 1 << ": ";

        cin >> requered[i];

    }

    cout << endl;

}

void nwcmethod::display()

{

    int i;

    cout << "\n==================== MATRIX ====================\n\n";

    cout << setw(9);

    for (i = 0; i < no\_of\_columns; i++)

        cout << "D" << i + 1 << setw(4);

    cout << setw(5) << "Supply" << endl << setw(0);

    for (i = 0; i < no\_of\_rows; i++)

    {

        cout << setw(3) << "S" << i + 1;

        for (int j = 0; j < no\_of\_columns; j++)

            cout << setw(5) << data[i][j];

        cout << setw(5) << capacity[i] << endl;

    }

    cout << setw(4) << "Demand";

    for (i = 0; i < no\_of\_columns; i++)

        cout << setw(5) << requered[i];

    cout << "\n\n==================== Solution ==================== \n\n";

    for (i = 0; i < no\_of\_rows; i++)

    {

        for (int j = 0; j < no\_of\_columns; j++)

        {

            if (allocation[i][j] != 0)

                cout << "\t" << data[i][j] << "\*" << allocation[i][j];

            else

                cout << "\t" << data[i][j];

        }

        cout << endl;

    }

    cout << endl;

    cout << "==================== ANSWER ====================\n\n";

    int k = 0, sum = 0;

    for (i = 0; i < no\_of\_rows; i++)

    {

        for (int j = 0; j < no\_of\_columns; j++)

        {

            if (allocation[i][j] != 0)

            {

                cout << "(" << data[i][j] << "\*" << allocation[i][j] << ")";

                if (k < no\_of\_allocation - 1)

                {

                    cout << " + ";

                    k++;

                }

                sum += data[i][j] \*allocation[i][j];

            }

        }

    }

    cout << "\n\nAnswer ===> " << sum;

    if ((no\_of\_rows + no\_of\_columns - 1) == no\_of\_allocation)

    {

        cout << "\n\nThis is a Non-Degenerated Solution.";

    }

    else

    {

            cout << "\n\nThis is a Degenerated Solution.";

    }

}

void nwcmethod::getData()

{

    cout << "==================== Matrix Values ====================" << endl<<endl;

    for (int i = 0; i < no\_of\_rows; i++)

    {

        cout << "Enter Elements for Row " << i+1 << ": ";

        for (int j = 0; j < no\_of\_columns; j++)

        {

            cin >> data[i][j];

        }

    }

    cout << endl;

}

int main()

{

    //clrscr();

    nwcmethod m1;

    int r, c;

    cout << "==================== Matrix Setup ==================== " << endl << endl;

    cout << "Enter No of Rows: ";

    cin >> r;

    cout << "Enter No of Columns: ";

    cin >> c;

    cout << endl;

    m1.setColumn(c);

    m1.setRow(r);

    m1.getData();

    m1.getCapacity();

    m1.getRequiredValue();

    m1.makeAllocation();

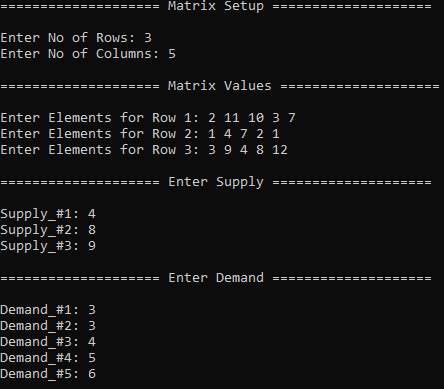
    // clrscr();

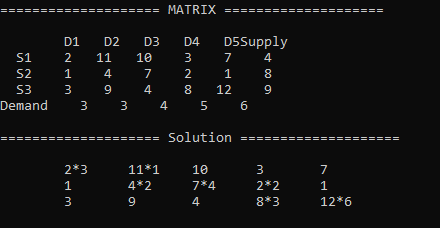
    m1.display();

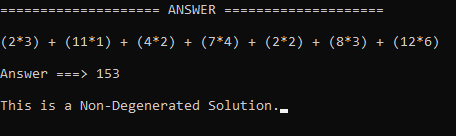
    getch();

}

### **Output**







# **Markov’s Chain - (C++ Language)**

### **Code**

#include <iostream>

using namespace std;

#define n 3

bool checkMarkov(double m[][n])

{

    for (int i = 0; i <n; i++) {

        double sum = 0;

        for (int j = 0; j < n; j++){

        sum = sum + m[i][j];}

        if (sum != 1)

        return false;

    }

    return true;}

int main(){

    double m[3][3];

    for(int i=0; i<3;i++){

        cout << "================ Enter Values for Row #" << i+1 <<" ================" <<endl;

        cout << "\nValues: ";

        for(int j=0; j<3; j++){

            cin >> m[i][j];}

        cout << endl;}

    cout << "================ MATRIX ================ " <<endl;

    cout <<"\t";

     for(int i=0; i<3;i++)

    {

        for(int j=0; j<3; j++)

        {

            cout << m[i][j] <<"\t";

        }

        cout << endl;

        if(i!=2)

        {

        cout <<"\t";

        }

    }

    cout << "================ ANSWER ================ " <<endl;

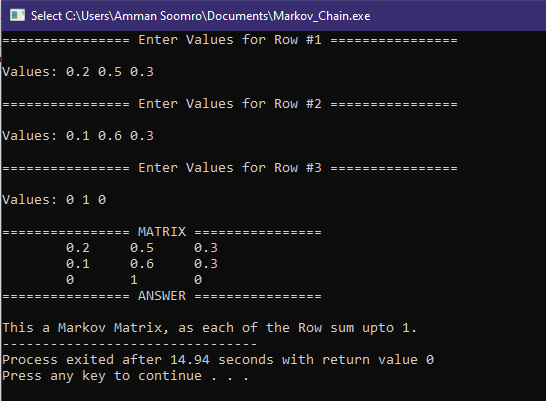
    if (checkMarkov(m)){

        cout << "\nThis a Markov Matrix, as each of the Row sum upto 1.";}

    else{

        cout << "\nThis is not a Markov Matrix.";}}

### Output



# **LPP Solver with Optimal Solution (Ptyhon Language)**

### Code

### Output

