**שפות תכנות מונחי עצמים**

**תרגיל תכנות מספר 3**

עליך לממש מחלקה ג'נרית בשם Matrix<T> המממשת מטריצה ריבועית בגודל nxn המשאירה את סוג המקדמים פתוח.

לשם כך עליך לתמוך במתודות הבאות:

1. מאפיין N לקריאה בלבד המחזיר את n.
2. בנאי

public Matrix(int n, bin\_op add, bin\_op sub, bin\_op mul)

 המקבל גודל מטריצה n ושמות של 3 מטודות המממשות פעולות חיבור, חיסור, כפל על המקדמים של המטריצה.

1. אינדקסר המקבל צמד אנדקסים [i,j] ומציב /מחזיר מקדם מ/אל המטריצה.
2. העמסת האופרטורים +, -, \* בינאריים על המטריצה. לחובבי האתגרים ניתן לממש את הכפל באלגוריתם של shtrassen.
3. מתודה print המדפיסה את איברי המטריצה.

יש להשתדל להגדיר את סוג האיברים לפי הסמל T כול אימת שאפשר.

לדוגמא, הפלט של התוכנית הבאה:

 class Class1

 {

 public static object int\_add(object x, object y)

 {

 int ix, iy, iz;

 ix = (int)x;

 iy = (int)y;

 iz = ix+iy;

 return ((object)iz);

 } // int\_add

 public static object int\_sub(object x, object y)

 {

 int ix, iy, iz;

 ix = (int)x;

 iy = (int)y;

 iz = ix-iy;

 return ((object)iz);

 } // int\_sub

 public static object int\_mul(object x, object y)

 {

 int ix, iy, iz;

 ix = (int)x;

 iy = (int)y;

 iz = ix\*iy;

 return ((object)iz);

 } // int\_mul

 public static object double\_add(object x, object y)

 {

 double dx, dy, dz;

 dx = (double)x;

 dy = (double)y;

 dz = dx+dy;

 return ((double)dz);

 } // double\_add

 public static object double\_sub(object x, object y)

 {

 double dx, dy, dz;

 dx = (double)x;

 dy = (double)y;

 dz = dx-dy;

 return ((double)dz);

 } // double\_sub

 public static object double\_mul(object x, object y)

 {

 double dx, dy, dz;

 dx = (double)x;

 dy = (double)y;

 dz = dx\*dy;

 return ((double)dz);

 } // double\_mul

 static void Main()

 {

 int n = 5, i, j;

 Matrix<double> M1d = new Matrix<double>(n,double\_add, double\_sub, double\_mul),

 M2d= new Matrix<double>(n,double\_add, double\_sub, double\_mul),

 M3d= new Matrix<double>(n,double\_add, double\_sub, double\_mul);

 double d;

 int k;

 for(i=0; i < M1d.N; i++)

 for(j=0; j < M1d.N; j++)

 {

 M1d[i,j] = 1.1\*i+j;

 M2d[i,j] = 10.1\*i + 1.1\*j;

 } // for

 Console.WriteLine("M1d:");

 M1d.print();

 Console.WriteLine("M2d:");

 M2d.print();

 Console.WriteLine("M1d + M2d:");

 M3d = M1d+M2d;

 M3d.print();

 M3d = M1d\*M2d;

 Console.WriteLine("M1d \* M2d:");

 M3d.print();

 d = (double)M1d;

 Console.WriteLine("\n(double)M1d = " + d);

 Matrix<int> M1i = new Matrix<int>(n,int\_add, int\_sub, int\_mul),

 M2i= new Matrix<int>(n,int\_add, int\_sub, int\_mul),

 M3i= new Matrix<int>(n,int\_add, int\_sub, int\_mul);

 for(i=0; i < M1i.N; i++)

 for(j=0; j < M1i.N; j++)

 {

 M1i[i,j] = i+10\*j;

 M2i[i,j] = 100\*i+10\*j;

 } // for

 Console.WriteLine("\n\nM1i:");

 M1i.print();

 Console.WriteLine("M2i:");

 M2i.print();

 Console.WriteLine("M1i + M2i:");

 M3i = M1i+M2i;

 M3i.print();

 M3i = M1i\*M2i;

 Console.WriteLine("M1i \* M2i:");

 M3i.print();

 k = (int)M1i;

 Console.WriteLine("\n(int)M1i = " + k);

 } // Main

 } // Class1

תהיה:

M1d:

 0 1 2 3 4

 1.1 2.1 3.1 4.1 5.1

 2.2 3.2 4.2 5.2 6.2

 3.3 4.3 5.3 6.3 7.3

 4.4 5.4 6.4 7.4 8.4

M2d:

 0 1.1 2.2 3.3 4.4

 10.1 11.2 12.3 13.4 14.5

 20.2 21.3 22.4 23.5 24.6

 30.3 31.4 32.5 33.6 34.7

 40.4 41.5 42.6 43.7 44.8

M1d + M2d:

 0 2.1 4.2 6.3 8.4

 11.2 13.3 15.4 17.5 19.6

 22.4 24.5 26.6 28.7 30.8

 33.6 35.7 37.8 39.9 42

 44.8 46.9 49 51.1 53.2

M1d \* M2d:

 303 314 325 336 347

 414.1 431.149999999999 448.2 465.25 482.3

 525.2 548.3 571.4 594.5 617.6

 636.3 665.45 694.6 723.75 752.9

 747.4 782.6 817.8 853 888.2

(double)M1d = 105

M1i:

 0 10 20 30 40

 1 11 21 31 41

 2 12 22 32 42

 3 13 23 33 43

 4 14 24 34 44

M2i:

 0 10 20 30 40

 100 110 120 130 140

 200 210 220 230 240

 300 310 320 330 340

 400 410 420 430 440

M1i + M2i:

 0 20 40 60 80

 101 121 141 161 181

 202 222 242 262 282

 303 323 343 363 383

 404 424 444 464 484

M1i \* M2i:

 30000 31000 32000 33000 34000

 31000 32050 33100 34150 35200

 32000 33100 34200 35300 36400

 33000 34150 35300 36450 37600

 34000 35200 36400 37600 38800

(int)M1i = 550