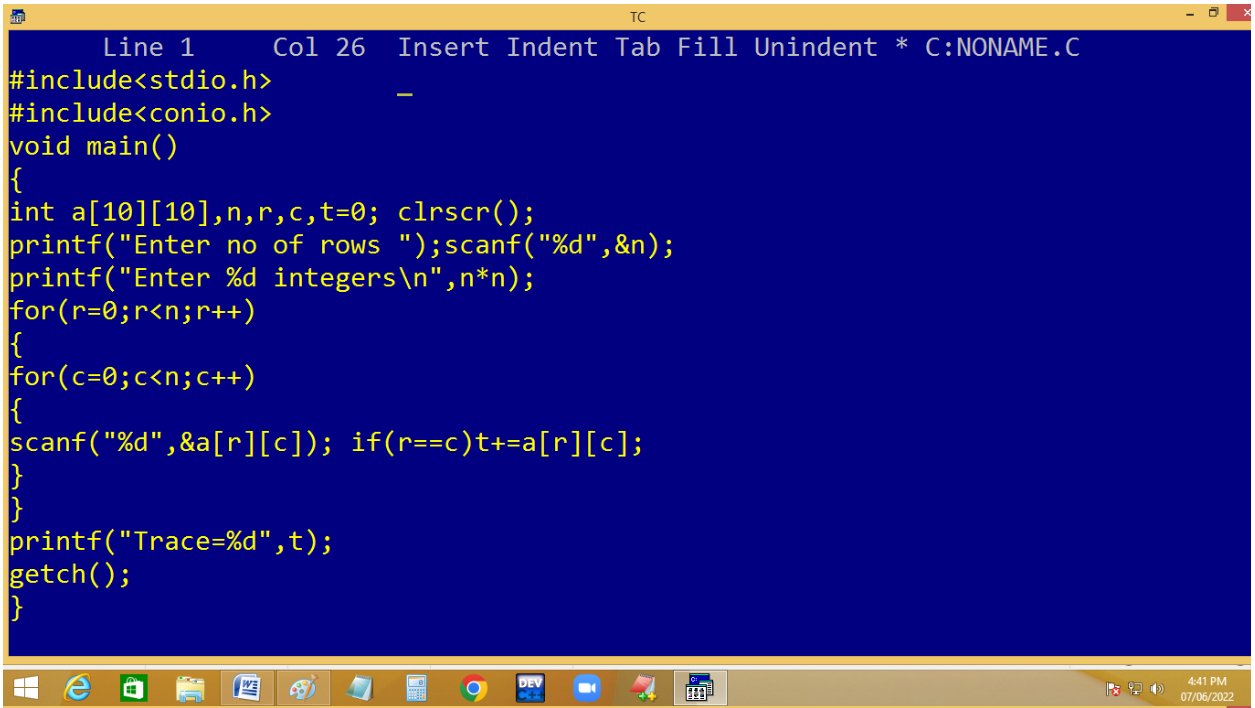
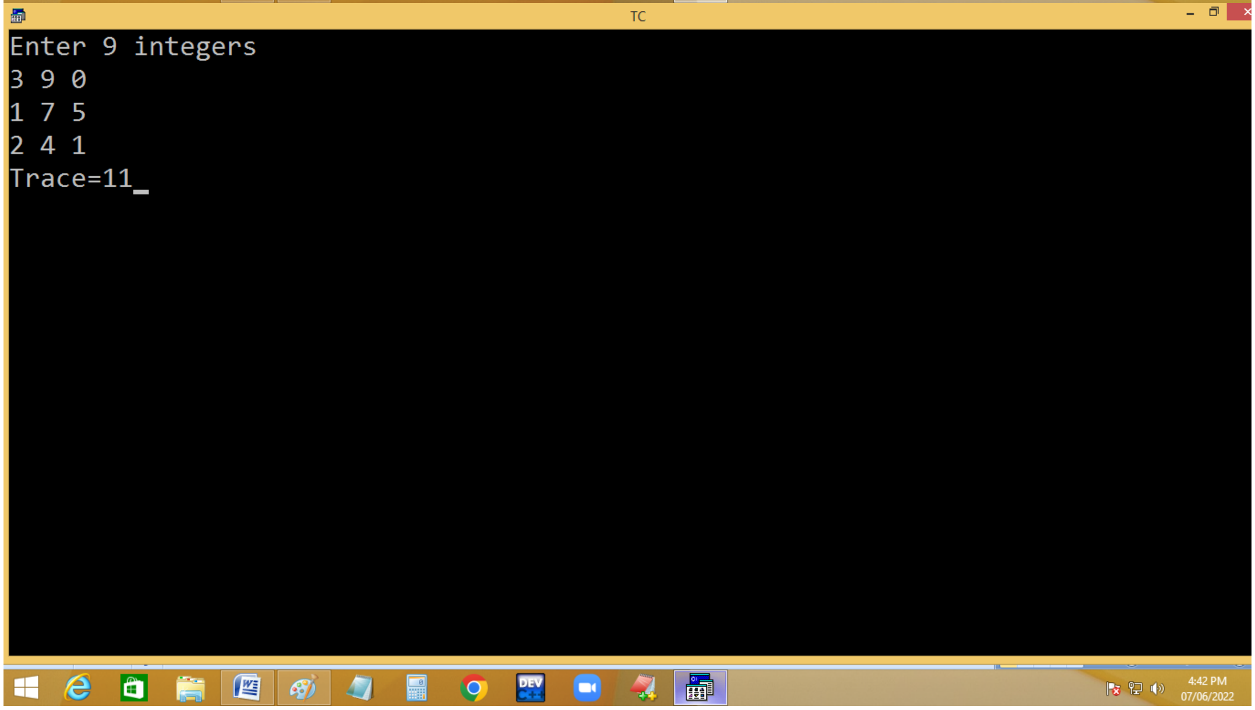
Finding trace of n*n matrix.

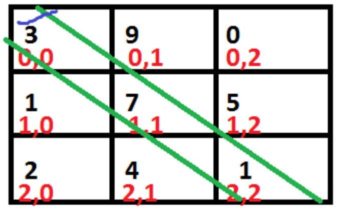


```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10][10],n,r,c,t=0; clrscr();
printf("Enter no of rows ");scanf("%d",&n);
printf("Enter %d integers\n",n*n);
for(r=0;r<n;r++)
{
for(c=0;c<n;c++)
{
scanf("%d",&a[r][c]); if(r==c)t+=a[r][c];
}
}
printf("Trace=%d",t);
getch();
}
```



```
Enter 9 integers
3 9 0
1 7 5
2 4 1
Trace=11_
```

```
for(r=0;r<3;r++)
{
for(c=0;c<3;c++)
{
scanf("%d",&a[r][c]);
if(r==c)t+=a[r][c];
}
}
p(t);
```



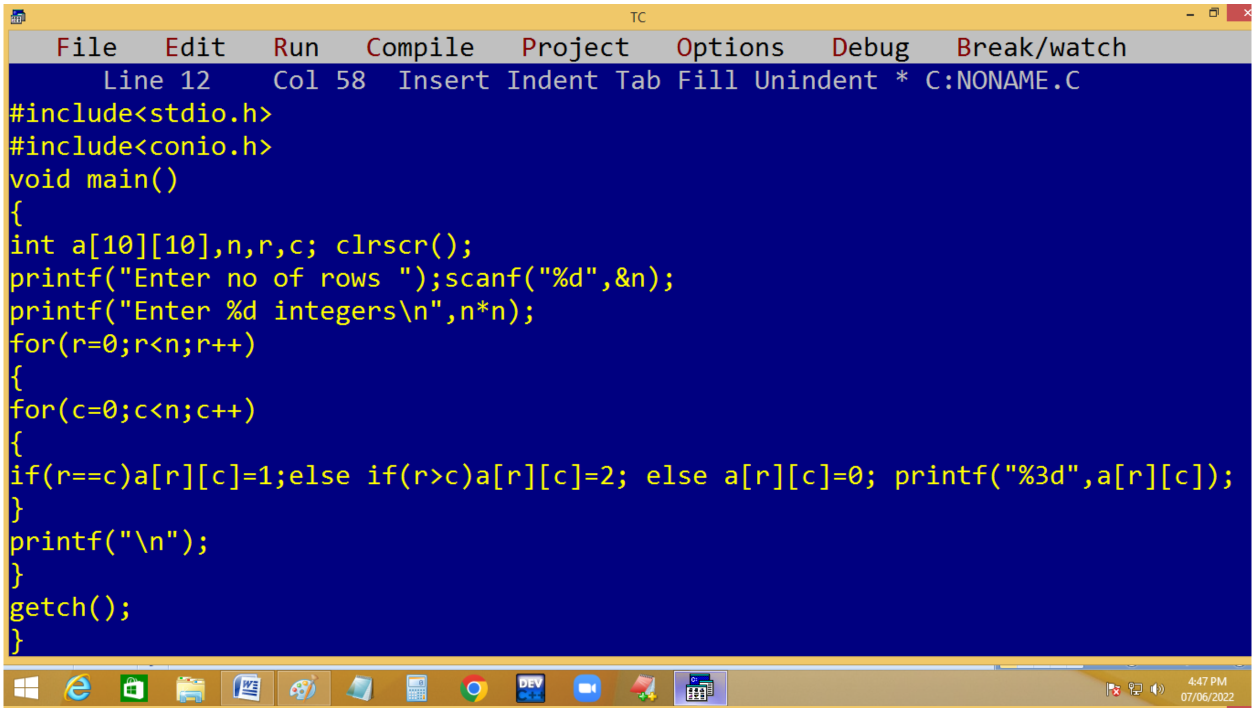
3 0,0	9 0,1	0 0,2
1 1,0	7 1,1	5 1,2
2 2,0	4 2,1	1 2,2

Trace = 11

r	c	T
0	0 1 2 ~~3~~	0+3=3
1	0 1 2 ~~3~~	3+7=10
2	0 1 2 ~~3~~	10+1 = 11
~~3~~		

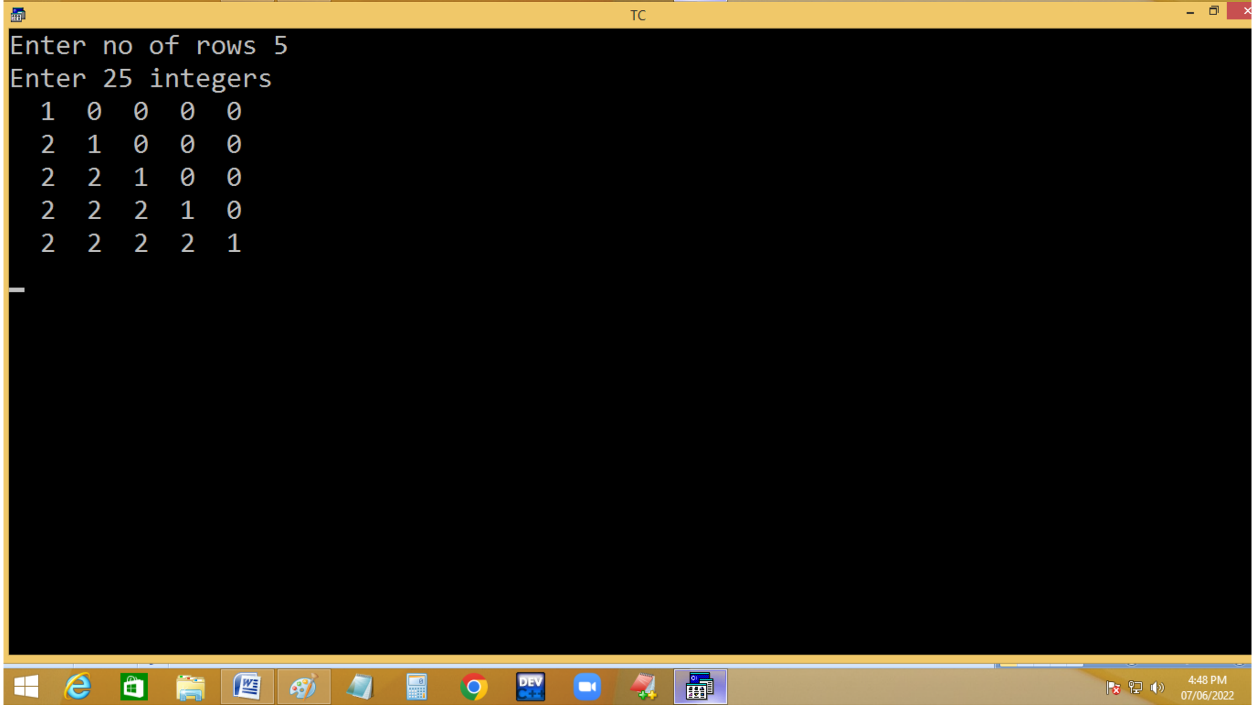
Eg. printing below output using a matrix.

```
1    0    0

2    1    0

2    2    1
```



TC
File Edit Run Compile Project Options Debug Break/watch
Line 12 Col 58 Insert Indent Tab Fill Unindent * C:NONAME.C
4:47 PM
07/06/2022

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10][10],n,r,c; clrscr();
printf("Enter no of rows ");scanf("%d",&n);
printf("Enter %d integers\n",n*n);
for(r=0;r<n;r++)
{
for(c=0;c<n;c++)
{
if(r==c)a[r][c]=1;else if(r>c)a[r][c]=2; else a[r][c]=0; printf("%3d",a[r][c]);
}
printf("\n");
}
getch();
}
```

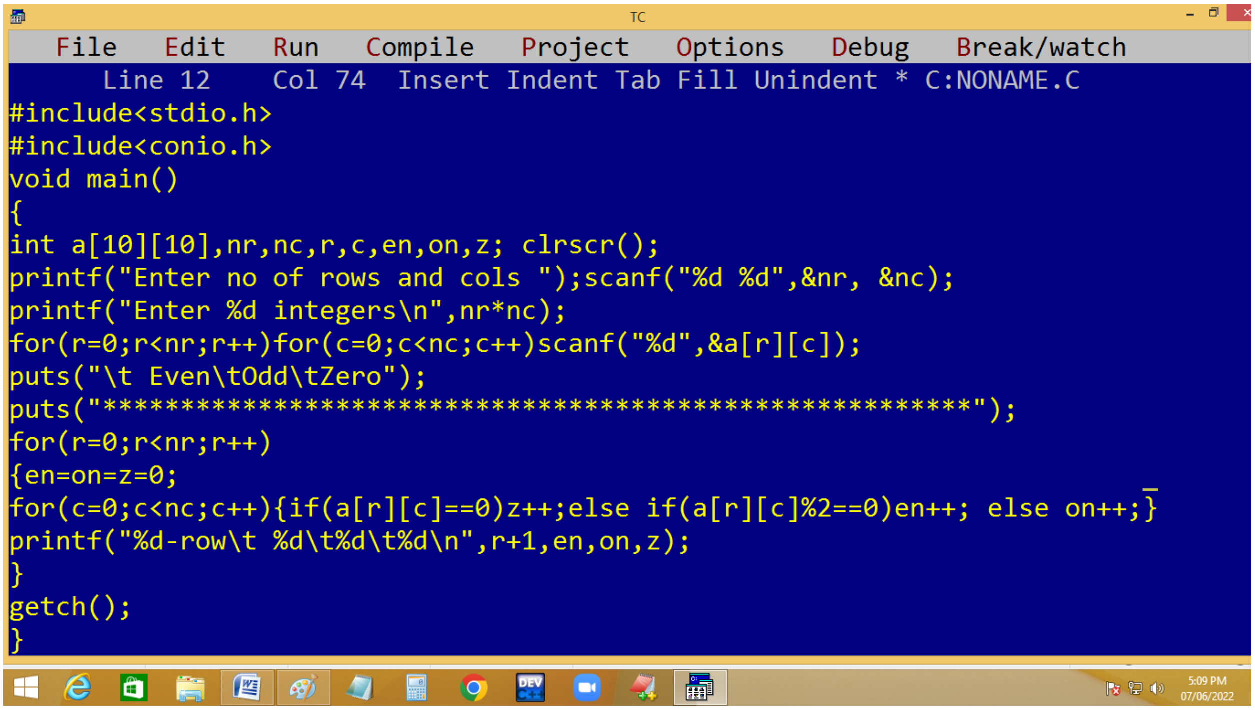


```
Enter no of rows 5
Enter 25 integers
  1  0  0  0  0
  2  1  0  0  0
  2  2  1  0  0
  2  2  2  1  0
  2  2  2  2  1
```

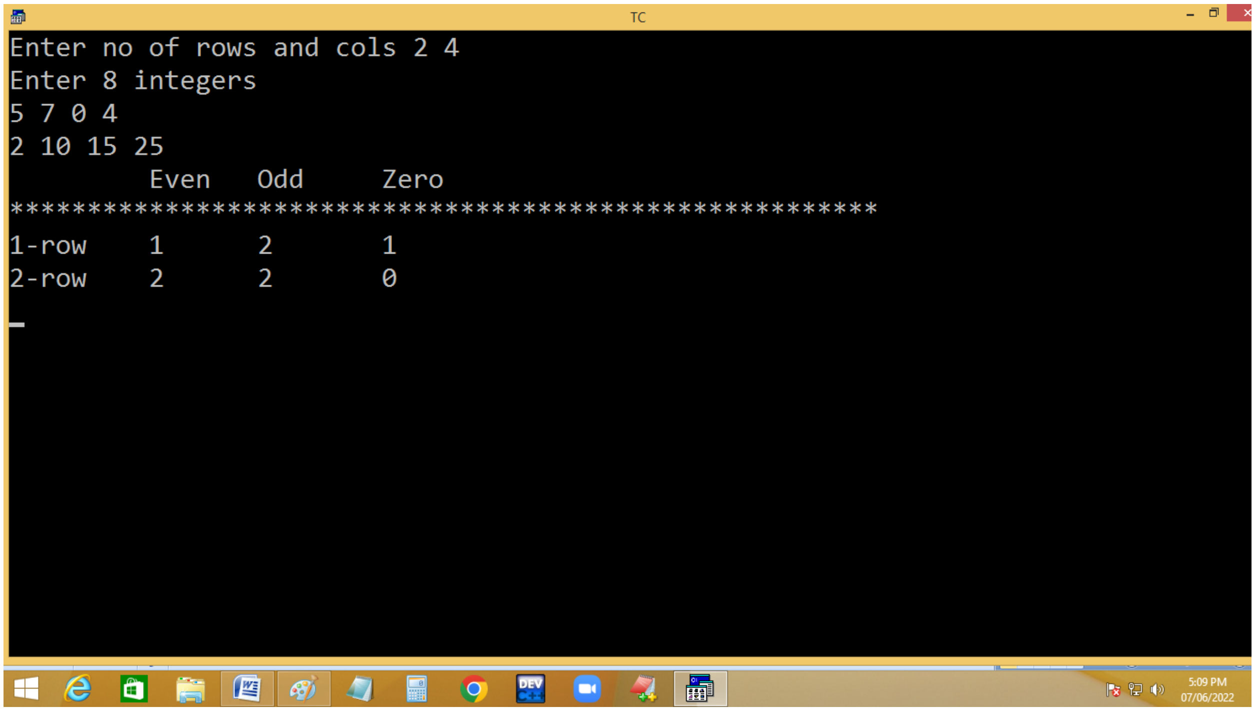
```
for(r=0;r<3;r++)
{
for(c=0;c<3;c++)
{
  if(r==c)a[r][c]=1;
  else if(r>c)a[r][c]=2;
  else  a[r][c]=0;
}
}
```

1 0,0	0 0,1	0 0,2
2 1,0	1 1,1	0 1,2
2 2,0	2 2,1	1 2,2

Eg. finding no of even, odd, zero in each row.



```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10][10],nr,nc,r,c,en,on,z; clrscr();
printf("Enter no of rows and cols ");scanf("%d %d",&nr, &nc);
printf("Enter %d integers\n",nr*nc);
for(r=0;r<nr;r++)for(c=0;c<nc;c++)scanf("%d",&a[r][c]);
puts("\t Even\tOdd\tZero");
puts("********************************************************");
for(r=0;r<nr;r++)
{en=on=z=0;
for(c=0;c<nc;c++){if(a[r][c]==0)z++;else if(a[r][c]%2==0)en++; else on++;}
printf("%d-row\t %d\t%d\t%d\n",r+1,en,on,z);
}
getch();
}
```



```
Enter no of rows and cols 2 4
Enter 8 integers
5 7 0 4
2 10 15 25
         Even    Odd      Zero
********************************************************
1-row    1       2        1
2-row    2       2        0
```

```
puts("\t Even\tOdd\tZero");
puts("-----------------------------------------------");
for(r=0;r<2;r++)
{
en=on=z=0; ✓
for(c=0;c<4;c++) ✓
{
if(a[r][c]==0)z++;
else if(a[r][c]%2==0)en++; else on++;
}
printf("%d-row\t%d\t%d\t%d\n",r+1,en,on,z);
}
```

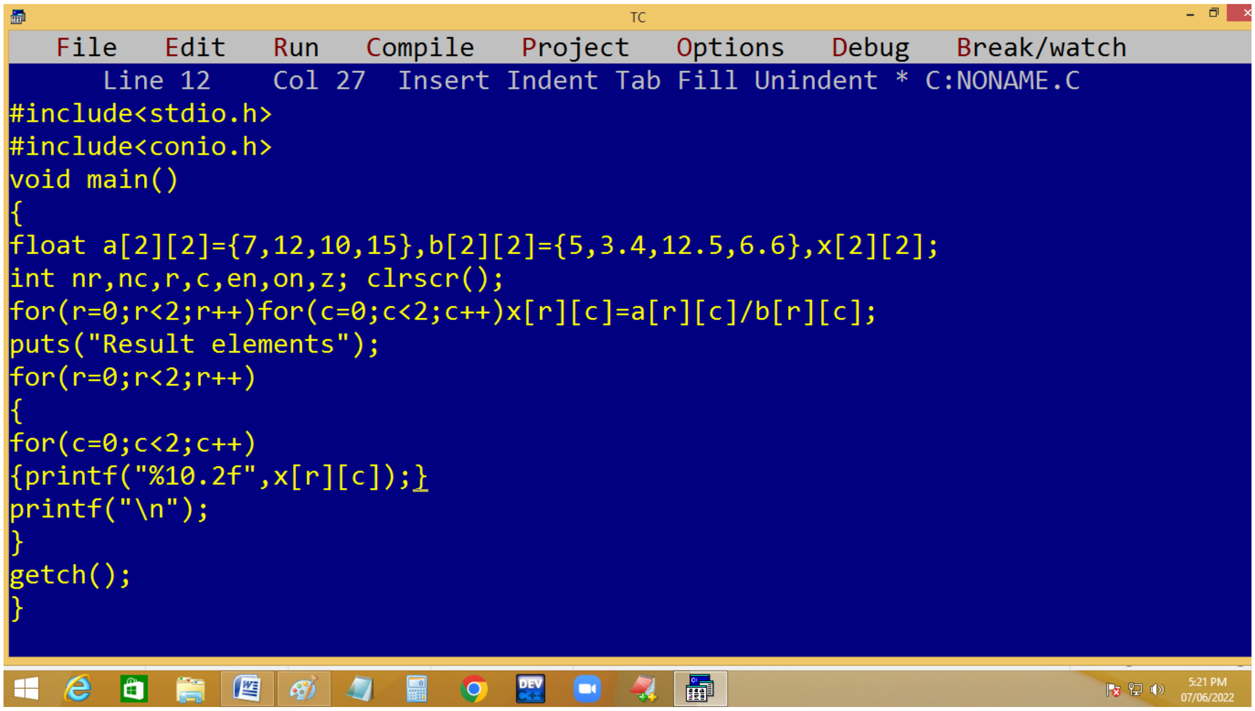
5 0,0	7 0,1	0 0,2	4 0,3
2 1,0	10 1,1	15 1,2	25 1,3

r	c
0	0123
1	0123
~~2~~	

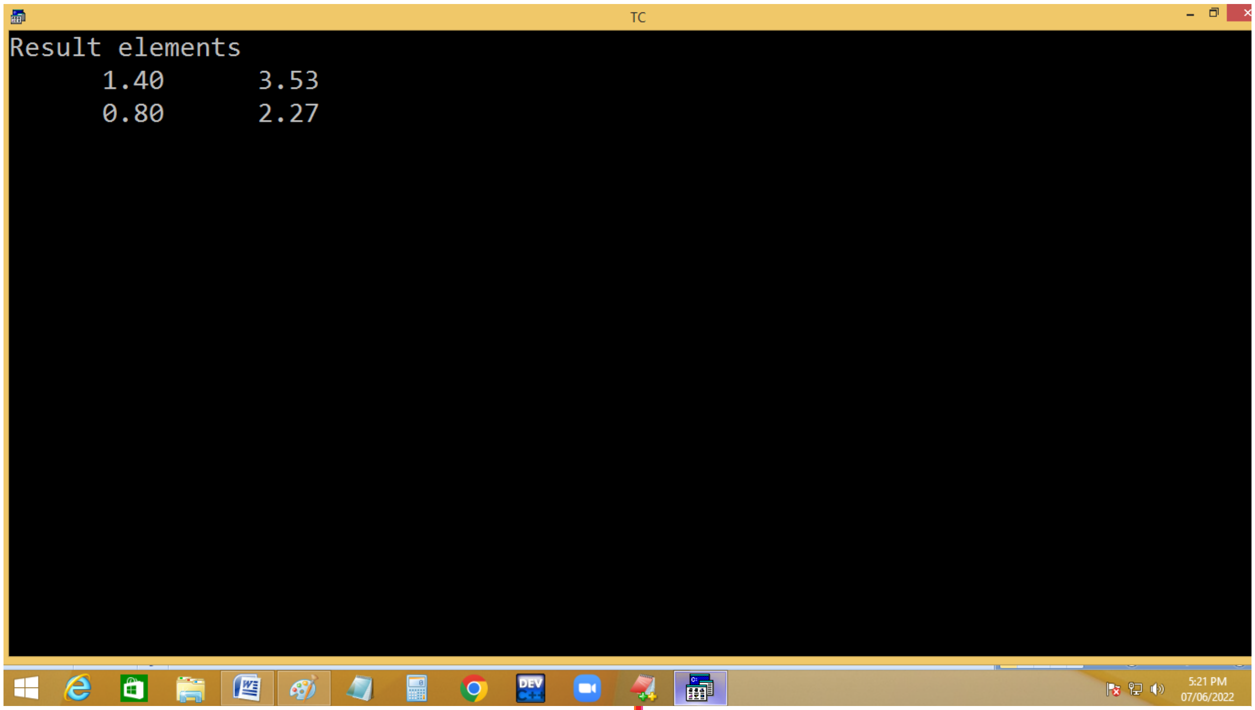
on	en	z
012	01	01
012	012	0

	Even	Odd	Zero
1-row	1	2	1 ✓
2-row	2	2	0

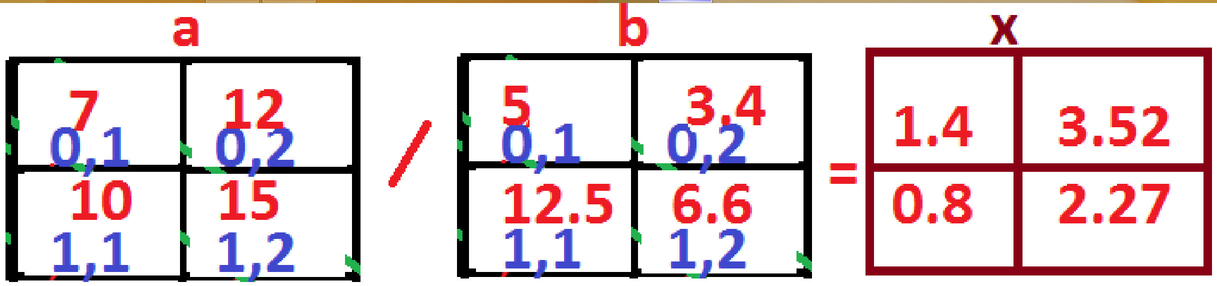
Eg. finding fractions of n*n matrix.



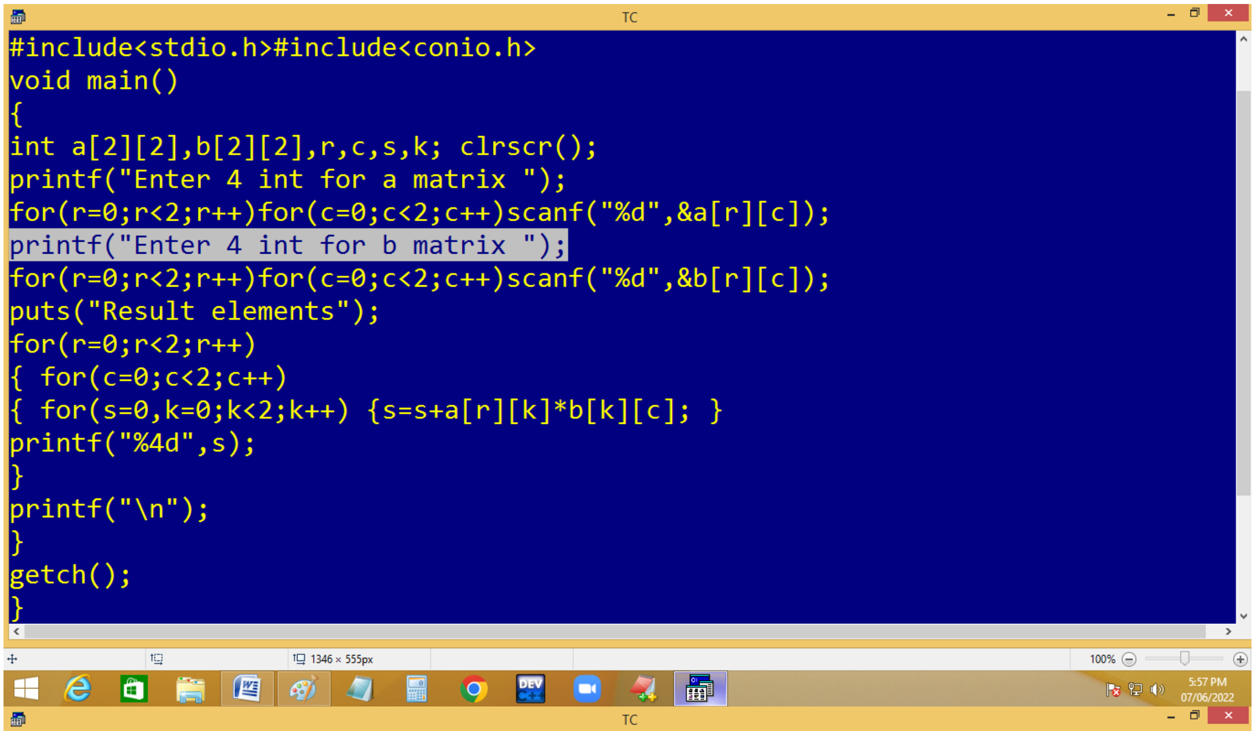
```
#include<stdio.h>
#include<conio.h>
void main()
{
float a[2][2]={7,12,10,15},b[2][2]={5,3.4,12.5,6.6},x[2][2];
int nr,nc,r,c,en,on,z; clrscr();
for(r=0;r<2;r++)for(c=0;c<2;c++)x[r][c]=a[r][c]/b[r][c];
puts("Result elements");
for(r=0;r<2;r++)
{
for(c=0;c<2;c++)
{printf("%10.2f",x[r][c]);}
printf("\n");
}
getch();
}
```



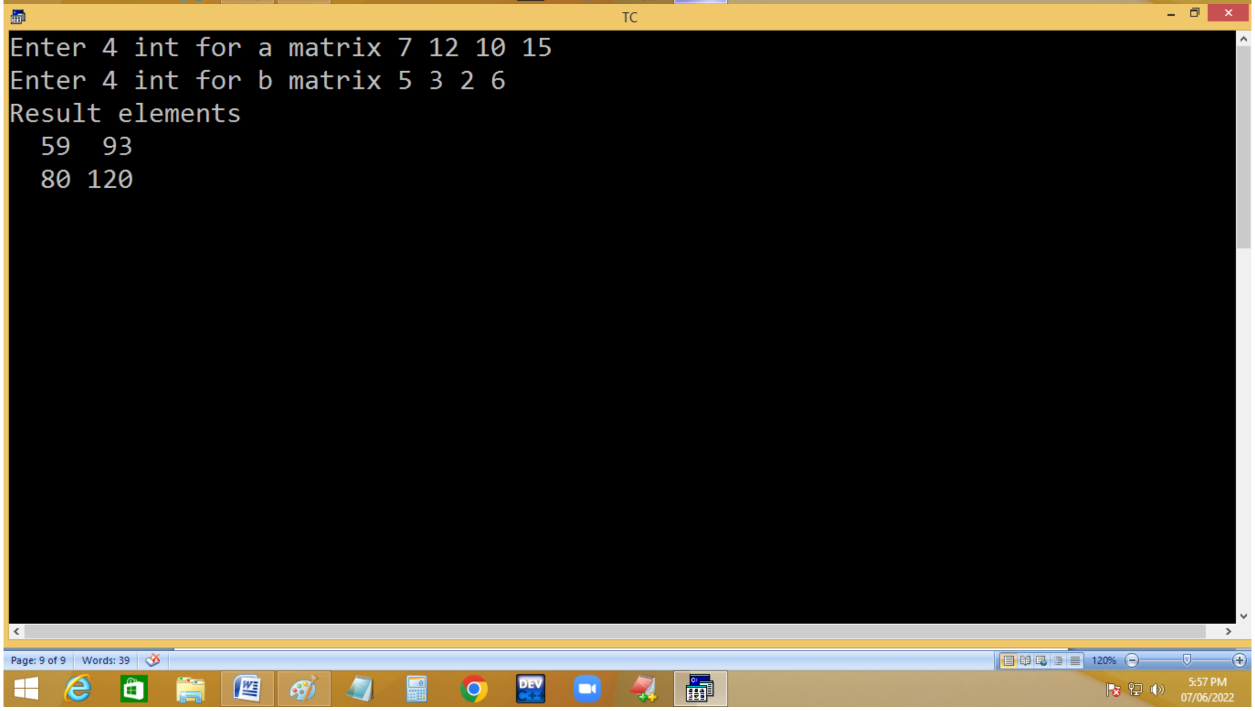
```
Result elements
      1.40      3.53
      0.80      2.27
```



Matrix multiplication:

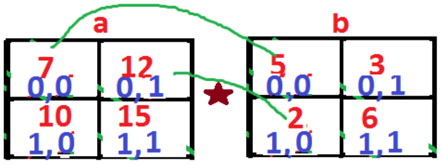


```
#include<stdio.h>#include<conio.h>
void main()
{
int a[2][2],b[2][2],r,c,s,k; clrscr();
printf("Enter 4 int for a matrix ");
for(r=0;r<2;r++)for(c=0;c<2;c++)scanf("%d",&a[r][c]);
printf("Enter 4 int for b matrix ");
for(r=0;r<2;r++)for(c=0;c<2;c++)scanf("%d",&b[r][c]);
puts("Result elements");
for(r=0;r<2;r++)
{ for(c=0;c<2;c++)
{ for(s=0,k=0;k<2;k++) {s=s+a[r][k]*b[k][c]; }
printf("%4d",s);
}
printf("\n");
}
getch();
}
```



```
Enter 4 int for a matrix 7 12 10 15
Enter 4 int for b matrix 5 3 2 6
Result elements
  59  93
  80 120
```

```
for(r=0;r<2;r++)
{
for(c=0;c<2;c++)
{ int s=0;
for(k=0;k<2;k++)
{
s=s+a[r][k]*b[k][c];
}
p(s);
}
p("\n");
}
```



7*5+12*2=59 7*3+12*6=93

10*5+15*2=80 10*3+15*6=120

r	c	k	s
0	0	012	0+7*5=35+12* 2=59
0	1	012	0+7*3=21+12*6=93
	2		
1	0	012	0+10*5=50+15*2=80
1	1	012	0+10*3=30+15*6=120
	2		
2			

3-dimensional arrays:

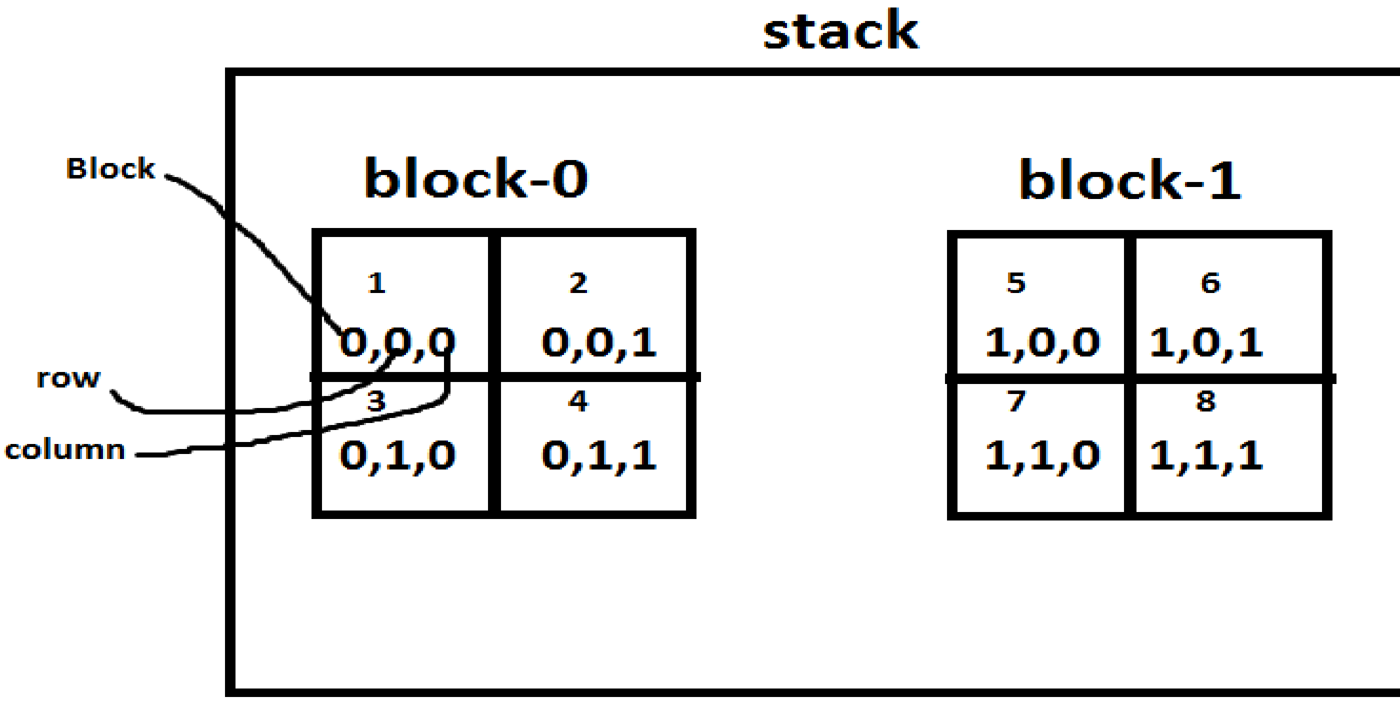
An array with several blocks, rows and columns.

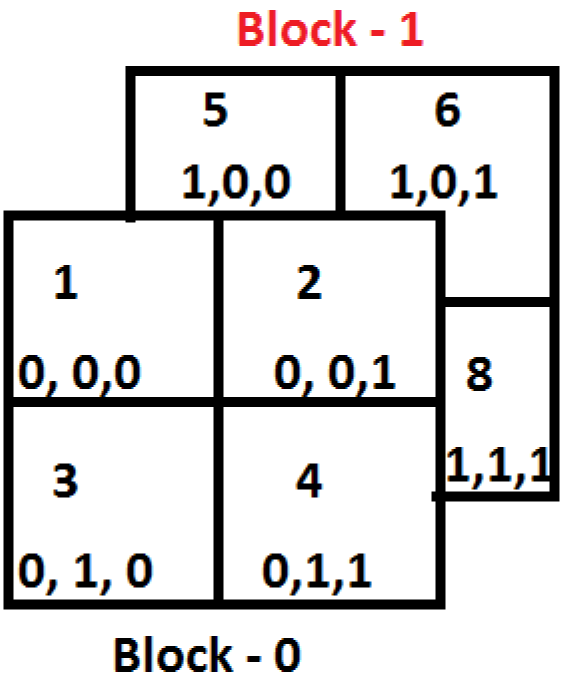
An array with 3 subscripting operators [ ] [ ] [ ].

Syntax:

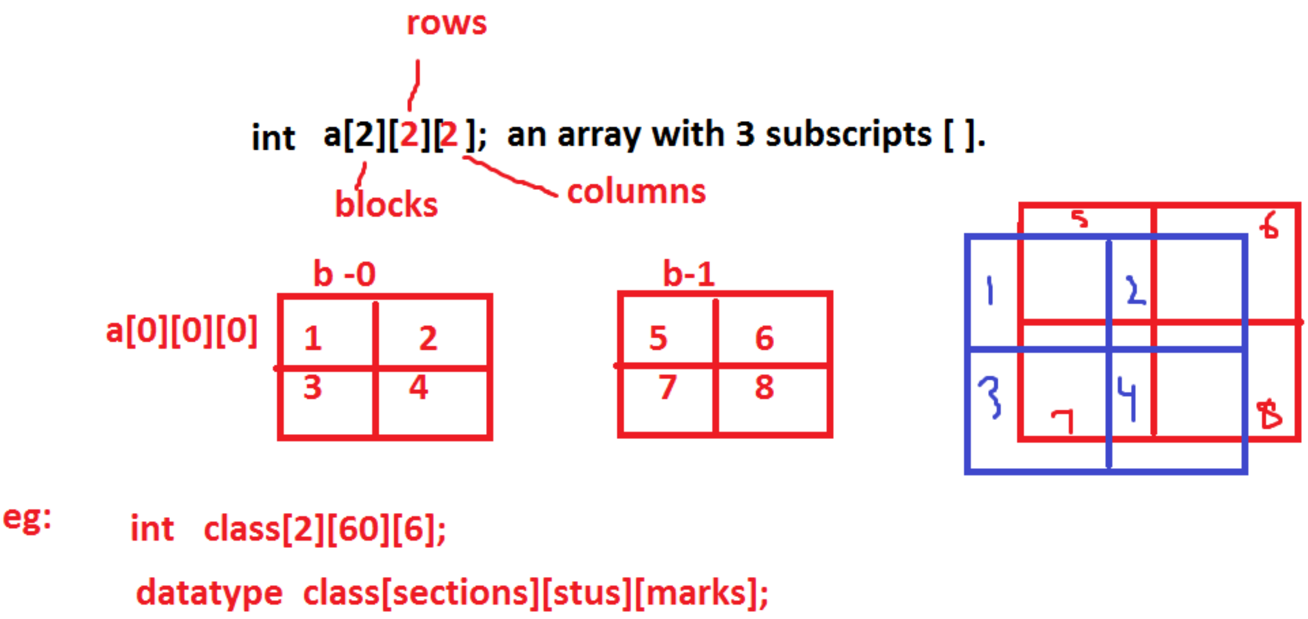
datatype variable [blocks] [rows] [columns];

Eg: int a[2][2][2]={1,2,3,4,5,6,7,8};


stack
Block
row
column
block-0
1
0,0,0
2
0,0,1
3
0,1,0
4
0,1,1
block-1
5
1,0,0
6
1,0,1
7
1,1,0
8
1,1,1


Block - 1
5
1,0,0
6
1,0,1
8
1,1,1
1
0, 0,0
2
0, 0,1
3
0, 1, 0
4
0,1,1
Block - 0

Eg:



```
int  a[2][2][2];  an array with 3 subscripts [ ].
```

eg:

```
int   class[2][60][6];
datatype  class[sections][stus][marks];
```

4-dimensional array:

An array with several sets, blocks, rows and columns.

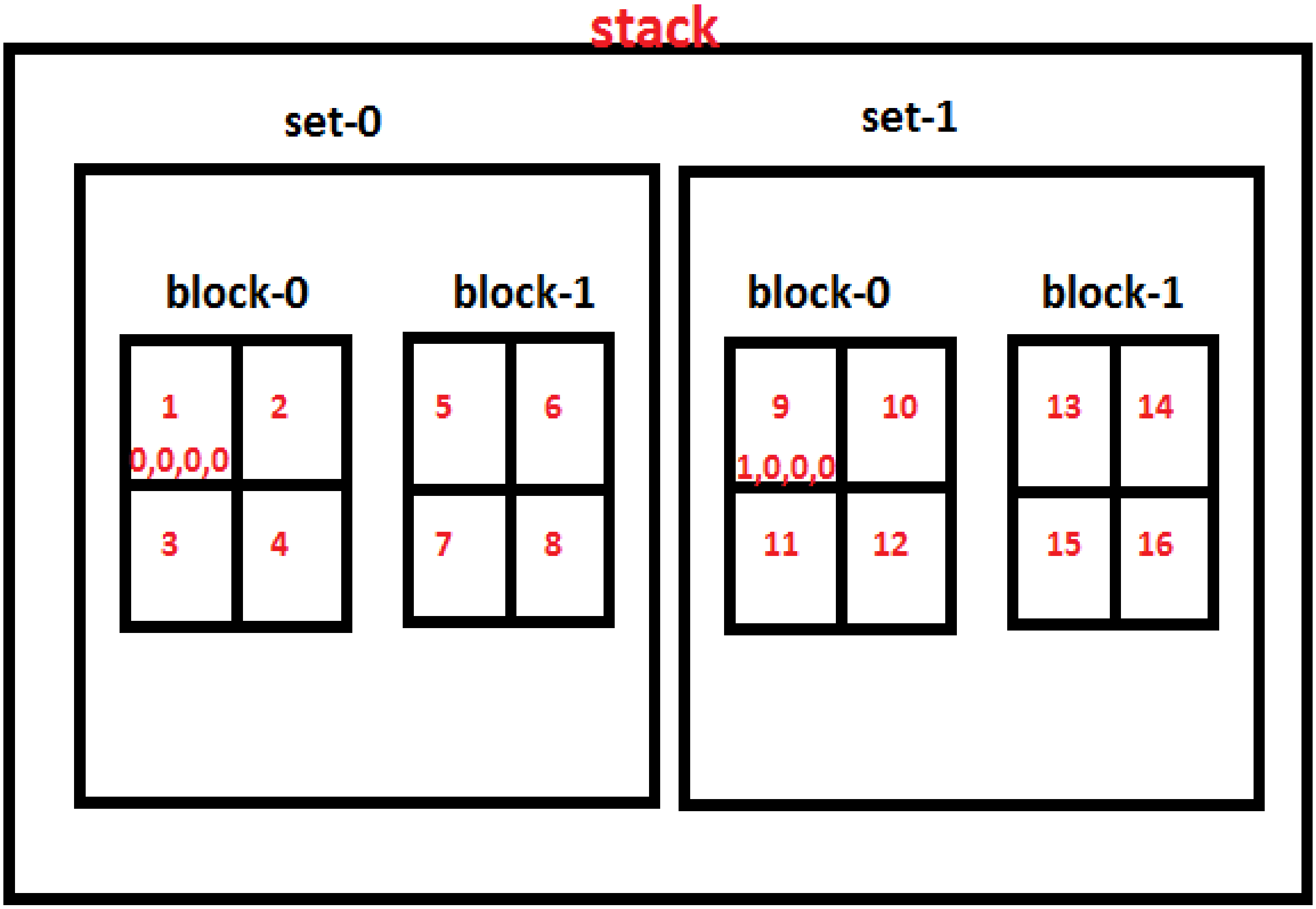
An array with 4 subscripting operators [ ] [ ] [ ] [ ] .

Syntax:

datatype variable [sets] [blocks] [rows] [cols];

eg:

```
int  a[2] [2] [2] [2]= {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
```



```
datatype   var[class][sec][stu][marks];
int        school[5][2][60][6];
```

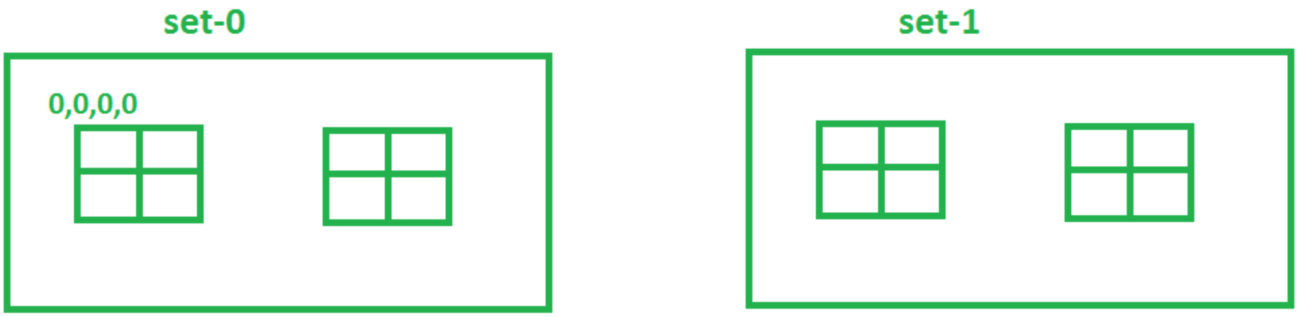
rows

set

int a[2][2][2][2]; an array with 3 subscripts [ ].

blocks

columns


set-0
set-1
0,0,0,0