








































# Practice Arena

Practice problems aimed to improve your coding skills.

-  PRACTICE-02\_SCAN-PRINT
-  PRACTICE-03\_TYPES
-  LAB-PRAC-02\_SCAN-PRINT
-  LAB-PRAC-01
-  PRACTICE-04\_COND
-  BONUS-PRAC-02
-  LAB-PRAC-03\_TYPES
-  PRACTICE-05\_COND-LOOPS
-  LAB-PRAC-04\_COND
-  LAB-PRAC-05\_CONDDLOOPS
-  PRACTICE-07\_LOOPS-ARR
-  LAB-PRAC-06\_LOOPS
-  LAB-PRAC-07\_LOOPS-ARR
-  LABEXAM-PRAC-01\_MIDSEM
-  PRACTICE-09\_PTR-MAT
-  LAB-PRAC-08\_ARR-STR
-  PRACTICE-10\_MAT-FUN
-  LAB-PRAC-09\_PTR-MAT
  -  Mr C writes a Story
  -  Matrix Arithmetic
  -  Spin the Matrix
  -  Crony Capitalization
  -  Matrix Mirroring
  -  Sodoku
  -  The Last Line
  -  Singular Value Decomposition
  -  Matrix Flip
  -  Now we are in Rome
  -  Search for the Submatrix
  -  Convoluted Convolutions
-  LAB-PRAC-10\_MAT-FUN
-  PRACTICE-11\_FUN-PTR
-  LAB-PRAC-11\_FUN-PTR
-  LAB-PRAC-12\_FUN-STRUC
-  LABEXAM-PRAC-02\_ENDSEM
-  LAB-PRAC-13\_STRUC-NUM
-  LAB-PRAC-14\_SORT-MISC

# Matrix Arithmetic

## LAB-PRAC-09\_PTR-MAT

## Matrix Arithmetic [20 marks]

---

### Problem Statement

Matrices possess a lot of the structures integers possess, some of which will be introduced to you in MTH102 and beyond. In this question we will look at two simple operations on square matrices. In the first line of the input you will be given 2 strictly positive integers  $m$  and  $p$ . In the next  $m$  lines you will be given the entries of an  $m \times m$  square integer matrix - let's call this matrix  $A$ . Each row of the matrix will be given in a different line with two elements separated by a single space.

You will have to output the matrices  $P = p \cdot A$  and  $Q = A^p$  in your output. The matrix  $p \cdot A$  is defined as the matrix each of whose entries is equal to  $p$  multiplied with the corresponding entry of  $A$  i.e.  $i,j$  entry of  $P = i,j$  entry of  $A$  times  $p$ . The matrix  $A^p$  is obtained as the iterated product  $(((((A \cdot A) \cdot A) \cdot A) \cdot A) \cdot A) \cdot A) \cdot A \dots \cdot A$  where  $A$  is multiplied with itself  $p$  times (since  $A$  is square, these products make sense dimensionally wise).

Print the matrices  $P$  and  $Q$  with each row printed on a different line with a single space between two elements of each row. There should be no trailing whitespaces at the end of any line.  $m$  and  $p$  will be smaller than or equal to 100 and all your outputs will fit inside the `int` datatype.

### Caution

1. Although  $m, p$  will always be strictly positive, the matrix entries may be zero or negative as well.
2. Matrix product is not commutative but it is associative i.e. if the product of three matrices  $A, B, C$  makes sense (i.e. dimensionally wise) then we always have  $(A * B) * C = A * (B * C)$
3. Be careful about extra/missing lines and extra/missing spaces in your output.

### HINTS:

1. You may try to compute the matrix exponential as a *running power* i.e. find  $A^1$  and then  $A^2$  and then  $A^3$  and so on.
2. In order to find the matrix exponential  $A^p$ , you have to organize your storage a bit. Use three matrices, one to store  $A$ , another to store the running power, and a third temporary matrix. Use the temporary matrix to compute and store the next power of  $A$ , given the previous power of  $A$  and then copy these values to the running power matrix.

### Code to manipulate matrices

```
int num[4][6] = {
    {1,2,3,4,5,6},
    {10,20,30,40,50,60},
    {100,200,300,400,500,600},
    {1000,2000,3000,4000,5000,6000}
};
int i,j;
for(i = 0; i < 4; i++){
    for(j = 0; j < 6; j++){
```

```
        printf("%d", num[i][j]);  
        if(j < 5) printf(" "); // No stray spaces at the end  
    }  
    if(i < 3) printf("\n"); // No stray new lines at the end  
}
```

---

**EXAMPLE:****INPUT**

```
2 2  
1 2  
2 1
```

**OUTPUT:**

```
2 4  
4 2  
5 4  
4 5
```

**Explanation:** The product matrix P is

```
2 4  
4 2  
and the matrix exponential is  
5 4  
4 5
```

---

**Grading Scheme:**

Total marks: **[20 Points]**

There will be partial grading in this question. There are will be  $2 * m$  lines in your output, depending on the value of  $m$ . Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

 **Start Solving!** (/editor/practice/6183)