```c
#include <stdio.h>

int main(){
    int m, p;
    scanf("%d%d", &m, &p);
    int a[m][m], res[m][m], temp[m][m], i, j, k;

    // Read in the matrix and also calculate P simultaneously
    for(i = 0; i < m; i++){
        for(j = 0; j < m; j++){
            scanf("%d", &a[i][j]);
            res[i][j] = p * a[i][j];
        }
    }

    // Print P right now
    for(i = 0; i < m; i++){
        for(j = 0; j < m; j++){
            printf("%d", res[i][j]);
            if(j < m - 1) printf(" "); // No trailing spaces
        }
        if(i < m) printf("\n"); // No trailing new lines
    }

    // A^1 = A
    for(i = 0; i < m; i++)
        for(j = 0; j < m; j++)
            res[i][j] = a[i][j];
    p--;

    // Loop invariant: at the beginning of the i-th iteration
    // res must contain A^i and the i-th iteration will simply
    // calculate A^{i+1} as A^i * A
    while(p){
        // temp = res * A
        // since by assumption, res contains A^i
        for(i = 0; i < m; i++){
            for(j = 0; j < m; j++){
                temp[i][j] = 0;
                for(k = 0; k < m; k++)
                    temp[i][j] += a[i][k] * res[k][j];
            }
        }
        // res = temp
        for(i = 0; i < m; i++)
            for(j = 0; j < m; j++)
                res[i][j] = temp[i][j];
        p--;
    }

    // Print A^p taking care not to have trailing spaces or newlines
    for(i = 0; i < m; i++){
        for(j = 0; j < m; j++){
            printf("%d", res[i][j]);
            if(j < m - 1) printf(" ");
        }
        if(i < m - 1) printf("\n");
    }

    return 0;
}
```