```
#include<stdio.h>
#include<omp.h>

#define SIZE 3

int main() {
#pragma omp parallel for collapse(2)
  for(int i=0; i<SIZE; ++i)
    for(int j=0; j<SIZE; ++j)
      printf("[%d,%d] by thread %d\n", i, j, omp_get_thread_num());
  return 0;
}
```

```
> gcc for_loop.c -fopenmp -o for_loop
> ./for_loop
[4,0] [4,1] [4,2] [4,3] [4,4] by thread 4
[1,0] [1,1] [1,2] [1,3] [1,4] by thread 1
[2,0] [2,1] [2,2] [2,3] [2,4] by thread 2
[3,0] [3,1] [3,2] [3,3] [3,4] by thread 3
[0,0] [0,1] [0,2] [0,3] [0,4] by thread 0
```

```
#include<stdio.h>
#include<omp.h>

#define SIZE 3

int main() {
#pragma omp parallel for collapse(2)
  for(int i=0; i<SIZE; ++i)
    for(int j=0; j<SIZE; ++j)
      printf("[%d,%d] by thread %d\n", i, j,
omp_get_thread_num());
  return 0;
}
```

```
> gcc for_loop.c -fopenmp -o for_loop
> ./for_loop
[0,0] by thread 0
[0,1] by thread 0
[1,0] by thread 2
[2,2] by thread 7
[1,2] by thread 4
[0,2] by thread 1
[1,1] by thread 3
[2,1] by thread 6
[2,0] by thread 5
```

```
#include<stdio.h>
#include<stdlib.h>
```

```
#include<omp.h>

#define M 300
#define N 400
#define P 300

int main() {
  int mat[M][P], mat1[M][N], mat2[N][P];
  for(int i=0; i<M; ++i)
    for(int j=0; j<P; ++j)
      mat[i][j] = 0;
  for(int i=0; i<M; ++i)
    for(int j=0; j<N; ++j)
      mat1[i][j] = rand()%10;
  for(int i=0; i<N; ++i)
    for(int j=0; j<P; ++j)
      mat2[i][j] = rand()%10;
  #pragma omp parallel for collapse(3)
  for(int i=0; i<M; ++i)
    for(int j=0; j<P; ++j)
      for(int k=0; k<N; ++k)
     mat[i][j] = mat1[i][k] * mat2[k][j];
  return 0;
}

> gcc matrix.c -fopenmp -o matrix
> /usr/bin/time ./matrix
parallel
0.37user 0.00system 0:00.37elapsed 677%CPU
serial
0.84user 0.00system 0:00.12elapsed 100%CPU
```