In [ ]:

```
code = """#include<iostream>

using namespace std;

__global__
void matrixVector(int *vec, int *mat, int *result, int n, int m)
{
    int tid = blockIdx.x*blockDim.x + threadIdx.x;
    int sum=0;

    if(tid <= n) {
        for(int i=0; i<n; i++) {
            sum += vec[i]*mat[(i*m) + tid];
        }
        result[tid] = sum;
    }
}

void init_array(int *a, int n) {
    for(int i=0; i<n; i++)
      a[i] = rand()%n + 1;
}

void init_matrix(int *a, int n, int m) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<m; j++) {
            a[i*m + j] = rand()%n + 1;
        }
    }
}

void print_array(int *a, int n) {
    for(int i=0; i<n; i++) {
        cout<<"  "<<a[i];
    }
    cout<<endl;
}

void print_matrix(int *a, int n, int m) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<m; j++)
          cout<<"  "<<a[i*m + j];
        cout<<endl;
    }
}

int main() {
    int *a, *b, *c;
    int *a_dev, *b_dev, *c_dev;

    int n = 3;
    int m = 4;

    a = new int[n];
    b = new int[n*m];
    c = new int[m];

    init_array(a, n);
    init_matrix(b, n, m);

    cout<<"Initial array : "<<endl;
    print_array(a, n);
    cout<<"Initial matrix : "<<endl;
    print_matrix(b, n, m);
    cout<<"Initial resultant array : "<<endl;
    print_array(c, m);
    cout<<endl;
    cudaMalloc(&a_dev, sizeof(int)*n);
    cudaMalloc(&b_dev, sizeof(int)*n*m);
    cudaMalloc(&c_dev, sizeof(int)*m);

    cudaMemcpy(a_dev, a, sizeof(int)*n, cudaMemcpyHostToDevice);
    cudaMemcpy(b_dev, b, sizeof(int)*n*m, cudaMemcpyHostToDevice);

    matrixVector<<<m/256+1, 256>>>(a_dev, b_dev, c_dev, n, m);

    cudaMemcpy(c, c_dev, sizeof(int)*m, cudaMemcpyDeviceToHost);

    cout<<"Results : "<<endl;
    print_array(c, m);

    cudaFree(a_dev);
    cudaFree(b_dev);
    cudaFree(c_dev);

    delete[] a;
    delete[] b;
    delete[] c;

    return 0;
}


"""
```

In [ ]:

```
text_file = open("matVec.cu", "w")
text_file.write(code)
text_file.close()
```

In [ ]:

```
!nvcc matVec.cu
```

In [ ]:

```
!./a.out
```

```
Initial array :
  2  2  1
Initial matrix :
  2  3  2  2
  1  1  2  3
  2  3  2  3
Initial resultant array :
  0  0  0  0

Results :
  0  0  0  0
```