1.Addition of vectors

```
#include <iostream>
using namespace std;

__global__ void add(int* A, int* B, int* C, int size) {
   int tid = blockIdx.x * blockDim.x + threadIdx.x;

   if (tid < size)
 {
      C[tid] = A[tid] + B[tid];
   }
}

void initialize(int* vector, int size) {
   for (int i = 0; i < size; i++) {
      vector[i] = rand() % 100;
   }
}

void print(int* vector, int size) {
   for (int i = 0; i < size; i++) {
      cout << vector[i] << " ";
   }
   cout << endl;
}

int main() {
   int N = 10;
   int* A, * B, * C;

   int vectorSize = N;
   size_t vectorBytes = vectorSize * sizeof(int);

   A = new int[vectorSize];
   B = new int[vectorSize];
   C = new int[vectorSize];

   initialize(A, vectorSize);
   initialize(B, vectorSize);

   cout << "Vector A: ";
   print(A, N);
   cout << "Vector B: ";
   print(B, N);

   int* X, * Y, * Z;
   cudaMalloc(&X, vectorBytes);
   cudaMalloc(&Y, vectorBytes);
   cudaMalloc(&Z, vectorBytes);
```

```
    cudaMemcpy(X, A, vectorBytes, cudaMemcpyHostToDevice);
    cudaMemcpy(Y, B, vectorBytes, cudaMemcpyHostToDevice);

    int threadsPerBlock = 256;
    int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;

    add<<<blocksPerGrid, threadsPerBlock>>>(X, Y, Z, N);

    cudaMemcpy(C, Z, vectorBytes, cudaMemcpyDeviceToHost);

    cout << "Addition: ";
    print(C, N);

    delete[] A;
    delete[] B;
    delete[] C;

    cudaFree(X);
    cudaFree(Y);
    cudaFree(Z);

    return 0;
}
```

Output:

```
Vector A: 83 86 77 15 93 35 86 92 49 21
Vector B: 62 27 90 59 63 26 40 26 72 36
Addition: 145 113 167 74 156 61 126 118 121 57
```

2. Matrix Multiplication

```
#include <cuda_runtime.h>
#include <iostream>

__global__ void matmul(int* A, int* B, int* C, int N) {
  int Row = blockIdx.y * blockDim.y + threadIdx.y;
  int Col = blockIdx.x * blockDim.x + threadIdx.x;
  if (Row < N && Col < N) {
    int Pvalue = 0;
    for (int k = 0; k < N; k++) {
      Pvalue += A[Row * N + k] * B[k * N + Col];
    }
    C[Row * N + Col] = Pvalue;
  }
}

int main() {
  int N = 512;
  int size = N * N * sizeof(int);
  int* A, * B, * C;
  int* dev_A, * dev_B, * dev_C;
  cudaMallocHost(&A, size);
  cudaMallocHost(&B, size);
  cudaMallocHost(&C, size);
  cudaMalloc(&dev_A, size);
  cudaMalloc(&dev_B, size);
  cudaMalloc(&dev_C, size);

  // Initialize matrices A and B
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      A[i * N + j] = i * N + j;
      B[i * N + j] = j * N + i;
    }
  }

  cudaMemcpy(dev_A, A, size, cudaMemcpyHostToDevice);
  cudaMemcpy(dev_B, B, size, cudaMemcpyHostToDevice);

  dim3 dimBlock(16, 16);
  dim3 dimGrid(N / dimBlock.x, N / dimBlock.y);

  matmul<<<dimGrid, dimBlock>>>(dev_A, dev_B, dev_C, N);

  cudaMemcpy(C, dev_C, size, cudaMemcpyDeviceToHost);

  // Print the result
  for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
      std::cout << C[i * N + j] << " ";
```

```
        }
        std::cout << std::endl;
    }

    // Free memory
    cudaFree(dev_A);
    cudaFree(dev_B);
    cudaFree(dev_C);
    cudaFreeHost(A);
    cudaFreeHost(B);
    cudaFreeHost(C);

    return 0;
}
```

Output:

```
44608256     111586048    178563840    245541632    312519424    379497216    446475008    513452800    580430592    647408384
111586048    312781568    513977088    715172608    916368128    1117563648   1318759168   1519954688   1721150208   1922345728
178563840    513977088    849390336    1184803584   1520216832   1855630080   -2103923968  -1768510720  -1433097472  -1097684224
245541632    715172608    1184803584   1654434560   2124065536   -1701270784  -1231639808  -762008832   -292377856   177253120
312519424    916368128    1520216832   2124065536   -1567053056  -963204352   -359355648   244493056    848341760    1452190464
379497216    1117563648   1855630080   -1701270784  -963204352   -225137920   512928512    1250994944   1989061376   -1567839488
446475008    1318759168   -2103923968  -1231639808  -359355648   512928512    1385212672   -2037470464  -1165186304  -292902144
513452800    1519954688   -1768510720  -762008832   244493056    1250994944   -2037470464  -1030968576  -24466688    982035200
580430592    1721150208   -1433097472  -292377856   848341760    1989061376   -1165186304  -24466688    1116252928   -2037994752
647408384    1922345728   -1097684224  177253120    1452190464   -1567839488  -292902144   982035200    -2037994752  -763057408
```