```
__global__ void vecAddKernel(float* d_A, float* d_B, float* d_C,
int n)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < n) C[i] = A[i] + B[i];
}
void vecAdd(float* A, float* B, float* C, int n)
{
    int size = n * sizeof (float); float* d_A, d_B, d_C;
    cudaMalloc((void **) &d_A, size);
    cudaMemcpy(d_A, A, size, cudaMemcpyHostToDevice);
    cudaMalloc((void **) &d_B, size);
    cudaMemcpy(d_B, B, size, cudaMemcpyHostToDevice);
    cudaMalloc((void **) &d_C, size);
    vecAddKernel<<<ceil(n/256.0), 256>>>(d_A, d_B, d_C, n);
    cudaMemcpy(C, d_C, size, cudaMemcpyDeviceToHost);
    cudaFree(d_A); cudaFree(d_B); cudaFree (d_C);
}
```