1 exercise 1

```
__global__ void vector_addition(int *A, int *B, size_t n) {
        A[threadIdx.x] += B[threadIdx.x];
}
```

2 Problem 2

question 1

```
__global__ void reverseArrayBlock(int *d_out, int *d_in)
{
// compute the offset of the block
    int inOffset = blockDim.x * blockIdx.x;
// compute the offset of the block after reversed
    int outOffset = blockDim.x * (gridDim.x - 1 - blockIdx.x);
//compute the offset of a thread in the original block
    int in = inOffset + threadIdx.x;
//compute the offset of a thread in the reversed block
    int out = outOffset + (blockDim.x - 1 - threadIdx.x);
    d_out[out] = d_in[in];
}
```

question 2

```
__global__ void reverseArrayBlock(int *d_out, int *d_in)
{
//declare shared array s_data;
    extern __shared__ int s_data[];
    int inOffset   = blockDim.x * blockIdx.x;
    int in  = inOffset + threadIdx.x;
    // Load one element per thread from device memory and store it
    // *in reversed order* into temporary shared memory
    s_data[blockDim.x - 1 - threadIdx.x] = d_in[in];
    // Block until all threads in the block have
    // written their data to shared mem
    __syncthreads();
    // write the data from shared memory in forward order,
    // but to the reversed block offset as before
    int outOffset = blockDim.x * (gridDim.x - 1 - blockIdx.x);
    int out = outOffset + threadIdx.x;
    d_out[out] = s_data[threadIdx.x];
}
```