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

```
// This is a kernel - it executes on the device (GPU), once per thread. Programmer's
// job is to write the kernel code from the standpoint of a single thread. The special
// variable 'threadIdx.x' is automatically provided - it tells you what your thread
// number is. In this example we call example_scalar_function on a single data array
// element.

__global__ my_kernel(float* d_data) {
      int tid = threadIdx.x;
      d_data[tid] = example_scalar_function(d_data[tid]);
}
```

```
// This is calling code that executes on the host (CPU). The purpose of this example
// function is to evaluate in parallel 'example_scalar_function' in place on every
// element of h_data.

void example(float* h_data, int data_size) {

  // Allocate memory on the device (GPU), and copy the data there.
  float* d_data;
  cudaMalloc((void **)&d_data, data_size*sizeof(float));
  cudaMemcpy(d_data, h_data, data_size*sizeof(float), cudaMemcpyHostToDevice);


  // Call kernel function.
  // This will create many threads and call 'example_kernel' in each thread.
  // In this example we create one thread for each element in the data array.
  int num_threads = data_size;
  my_kernel<<<1, num_threads>>>(d_data);


  // Copy back results from device memory to host memory, and free device memory.
  cudaMemcpy(h_data, d_data, data_size*sizeof(float), cudaMemcpyDeviceToHost);
  cudaFree(d_data);


  // CUDA exit - needed to flush printf write buffer.
  cudaDeviceReset();

}
```